Parveen Khurana

124 Followers About Following

Say Hi to Machine Learning

P Parveen Khurana Nov 27, 2019 · 23 min read

We have seen that there are **four challenges** that typically any **large scale rule-based system** actually faces:

i.) Lots of data

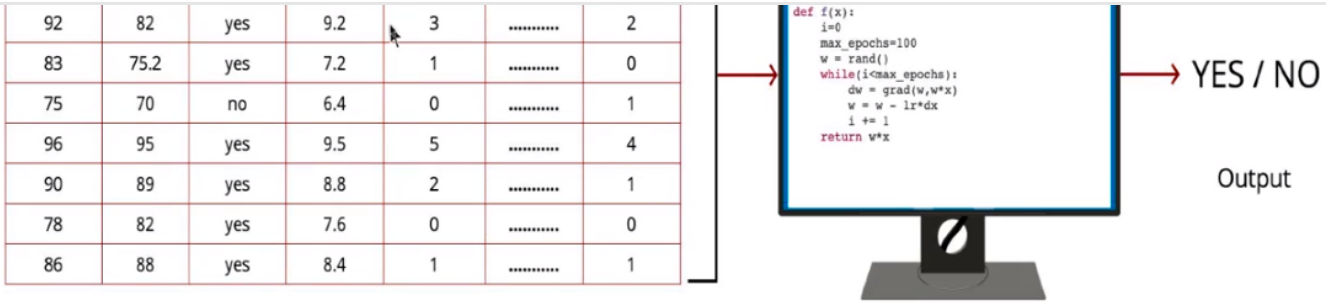
ii.) A lot of rules which are hard to write down

iii.) And sometimes the rules themselves are inexpressible we can't even write them as a Boolean condition.

iv.) Rules being unknown.

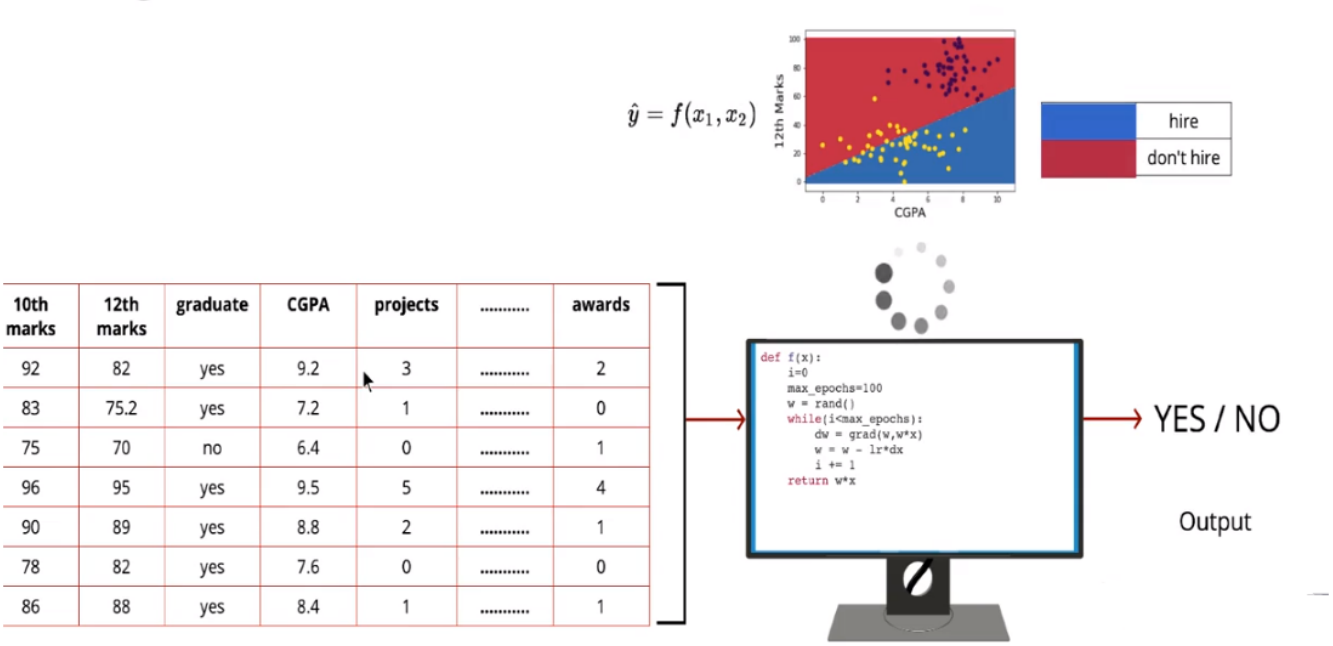
To overcome these limitations, we look at **Machine Learning**.

The idea here is that instead of having a human who comes with the rules(of expert-based systems) in their head and then tries to express these rules as programs, can we remove the human of the loop and introduce a machine that directly accesses this data and learns the program(function) on its own.



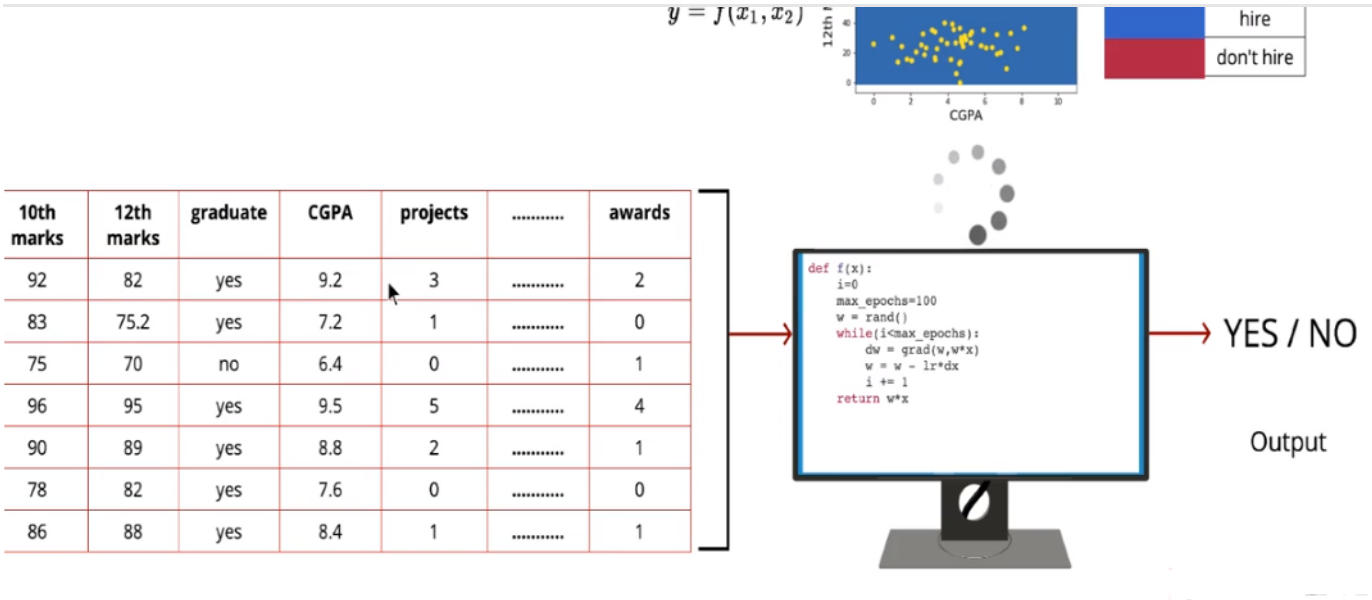
92	82	yes	9.2	3	.....	2
83	75.2	yes	7.2	1	.....	0
75	70	no	6.4	0	.....	1
96	95	yes	9.5	5	.....	4
90	89	yes	8.8	2	.....	1
78	82	yes	7.6	0	.....	0
86	88	yes	8.4	1	.....	1

So, instead of a human coding out the function/program, the machine has to somehow figure out this function, and not only that, it has to figure out some parameters as well.

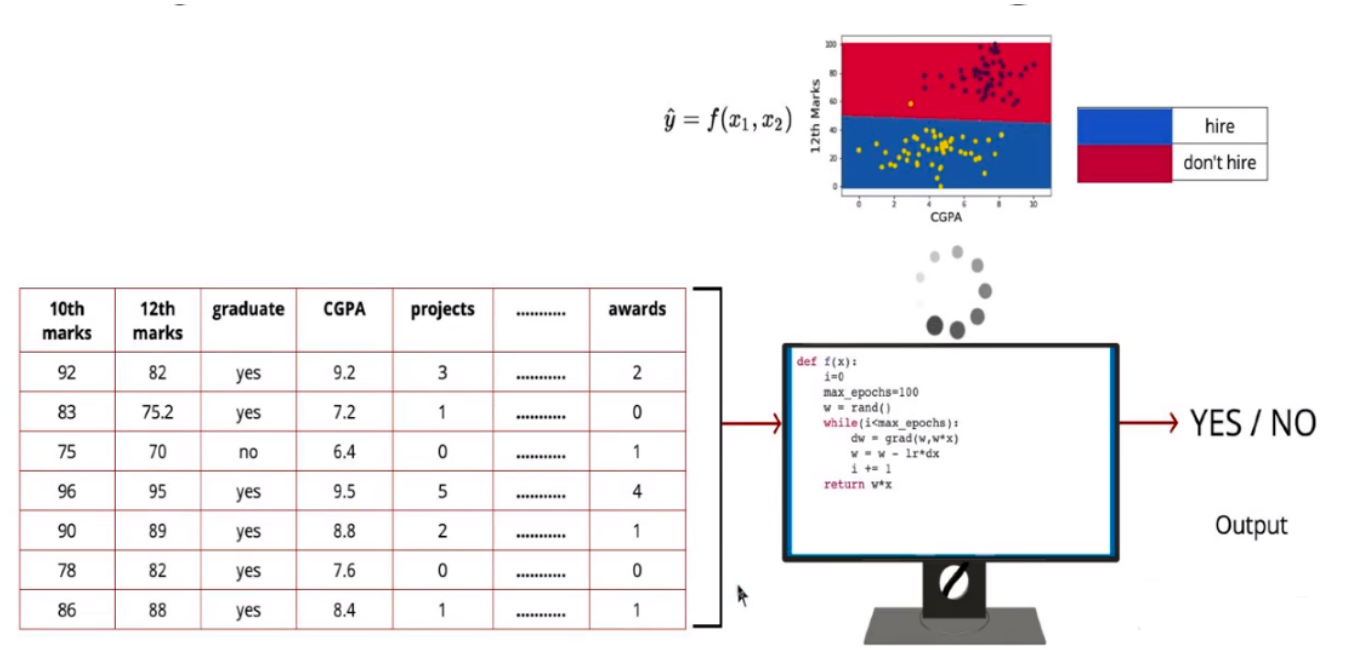


10th marks	12th marks	graduate	CGPA	projects	.....	awards
92	82	yes	9.2	3	.....	2
83	75.2	yes	7.2	1	.....	0
75	70	no	6.4	0	.....	1
96	95	yes	9.5	5	.....	4
90	89	yes	8.8	2	.....	1
78	82	yes	7.6	0	.....	0
86	88	yes	8.4	1	.....	1

Parameter set 1



Parameter set 2



Parameter set 3

So, this is what a typical machine learning set up looks like. We are given a lot of data. And now we have this function which depends on all of this data(all the rows all the features), and now our decision relies on this function, we just don't know what this function is, so unlike the other case where we were able to write this function as series of if-else conditions, here we don't even know what the function is. And unlike the other case where we were able to give certain weights to certain inputs(whether Java is

Now we can tell a machine that we think there is some relation between the input and the output, we think the relation comes from this family of functions(either the Linear family or polynomial family or from Non-Linear family, etc.) and we could just tell the machine, look at this data look at the inputs, look at the outputs, try out all these functions with different parameters and figure out which functions and with what parameters best explains the relationship between the input and the output.

So, this task is what we are outsourcing to the machine. And once the machine gives us that function, we just need to code up that function and for any new input, we could derive the output just by plugging in the input to that function. So, that's what Machine Learning enables us to do.

So, now the crux is how do you define these functions? We can't just try arbitrary functions, so what are the good set of functions to try and once the function is available, how do we estimate the values of these parameters.

Six Elements of Machine Learning:

i.) Data

ii.) Task

iii.) Model

iv.) Loss

v.) Learning Algorithm

vi.) Evaluation

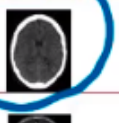
Data:

Data is the fuel of Machine Learning. Today we have data everywhere whether be it on social media platforms like Facebook where we have text data, video data, audio data or be it on product websites like Amazon where we have something known as structured data which is in form of a table or we have reviews or product picture or some videos to describe how to use the product and so on, or you could have something like gaana.com where we have a lot of audio or speech data. So, all forms of data are available on the internet. Just because so much data is available does not

application:

So, the application is here that we are given a medical image and based on that input we want to be able to train a machine to find out whether there is an anomaly or not in that image.

scans	x						y
	2.3	5.9	...	11.0	-0.3	8.9	0
	-8.5	-1.7	...	-1.3	9.0	7.2	1
	-0.4	6.7	...	-2.4	4.7	-7.3	0
	1.6	-0.4	...	-4.6	6.4	1.9	1
	3.9	-4.1	...	6.7	-3.1	2.1	1
	5.1	3.7	...	1.8	-4.2	9.3	1

scans	x						y
	2.3	5.9	...	11.0	-0.3	8.9	0
	-8.5	-1.7	...	-1.3	9.0	7.2	1
	-0.4	6.7	...	-2.4	4.7	-7.3	0
	1.6	-0.4	...	-4.6	6.4	1.9	1
	3.9	-4.1	...	6.7	-3.1	2.1	1
	5.1	3.7	...	1.8	-4.2	9.3	1

characteristic, in this case, it tells whether the input has an anomaly or not(maybe like brain tumor or any other sort of medical anomaly). So, for this, **we need data of the type (x, y) where x is the input and y is the output.** Only if we have abundant data of form (x, y) then only we can train a machine to learn a mapping between x and y. And then if we are given a new image, we should be able to predict the outcome for this. But if the only x is being provided, then there is not much that we can do.

And if the input contains an anomaly, then we would also like to locate the exact position of the anomaly example:

	-8.5	-1.7	...	-1.3	9.0	7.2	1
	-0.4	6.7	...	-2.4	4.7	-7.3	0
	1.6	-0.4	...	-4.6	6.4	1.9	1
	3.9	-4.1	...	6.7	-3.1	2.1	1
	5.1	3.7	...	1.8	-4.2	9.3	1

And y**, in this case, output would not be just 0 or 1, it would the co-ordinates of the bottom left corner and top right corner.** And for this particular case, we would require the data in this form where we are provided with samples(image as the input) for which we have the coordinates of the bounding box(bottom left corner and top right corner) where the anomaly lies so that we can train the machine accordingly and the machine's job is to find the relationship between the input and the coordinates of the bounding box which would be the output in this case.

The second thing that we need to understand is that the data should be in a machine-readable format. So, in the case of an image, it's not very difficult, the image(medical scan) could be in the form of a 30X30 image which means it has 900 pixels, each of the pixels tells the RGB value of that particular cell, so we have 900 values which we enumerate as a single array, so that's what out input is and what we are trying to learn is a relationship which takes you from this input of 900 numbers to the output number.

i.) Data should be in the form of (x, y).

ii.) It should be in a machine-readable format(All the data is encoded as numbers).

And usually, the data is high dimensional, in this case, we have 30X30 image or 900 pixels, that means we have 900 numbers describing each data point/sample.

Let's take another example where we have a document and we would like to do sentiment analysis on that document to know whether the reviewer is talking something positive about the product or something negative about the product:

Document	x						y
Don't buy this MI 6 Pro, Speaker volume is very bad	1.9	3.2	...	-9.8	-6.7	1.2	*negative*
Delivered as shown. Good price and fits perfect	1.3	3.6	...	-5.4	9.1	2.3	*positive*
What a phone.. A handy epic phone. MI at its best ...	0.4	7.6	...	-0.1	-1.4	8.7	*positive*
Its look stunning in pictures , but not in real.	1.5	-0.8	...	7.8	8.4	0.3	*negative*
Amazing camera and battery. Good deal!	2.5	-5.7	...	0.9	5.3	-8.1	*positive*

So, here again, **we need data in the form of (x, y) where x is the document and y is the label marked by a human**. Now if have a lot of such data, we could build a machine learning agent to understand the relation between x and y such that if you give me a new document as the input at the test time, our agent would be able to output y using the function it has learned.

Secondly, in this case, as we have text data, we should have a way to represent this data in some form of numerical quantity because the function that we are going to learn is of the form

$y = f(x)$

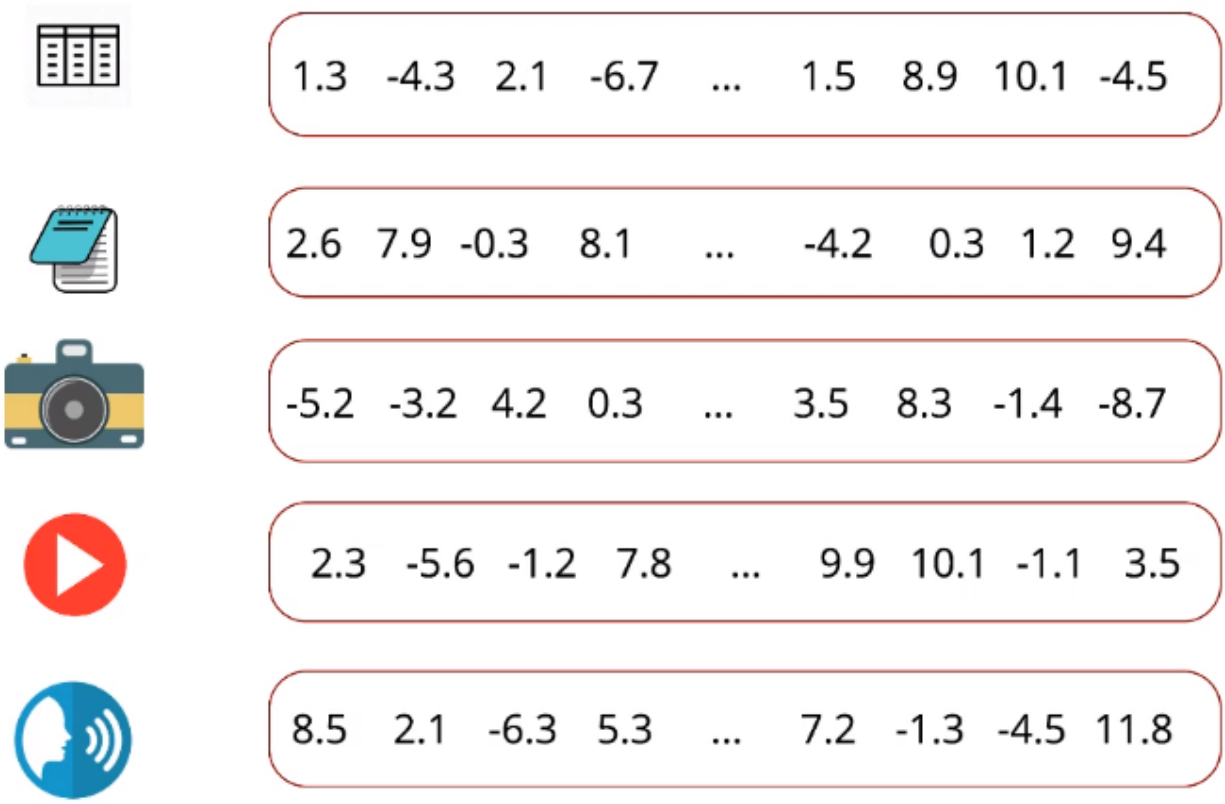
numbers.

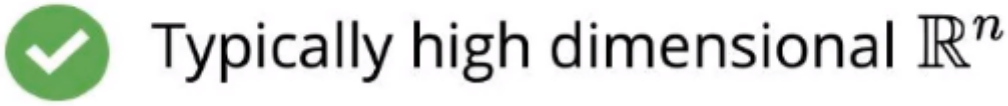
And the data just need not be text or image, it could a structured table, for example, it could be the past record of an employee like what was his appraisal ratings for years, how many teams did he work with, how many projects did we work on, what was his client satisfaction for every module he delivered, what was his salary at different levels, how many promotions he/she has got etc. So, all of these can be represented as structured data.

We could have text data, product reviews discussed above is an example.

Then we have the image data where we take the pixel values to represent it as numeric data. Even a video could be represented as numeric data as the video is just a collection of images/pictures.

And lastly, we have the speech data which we could represent as of different amplitudes, frequencies and so on.





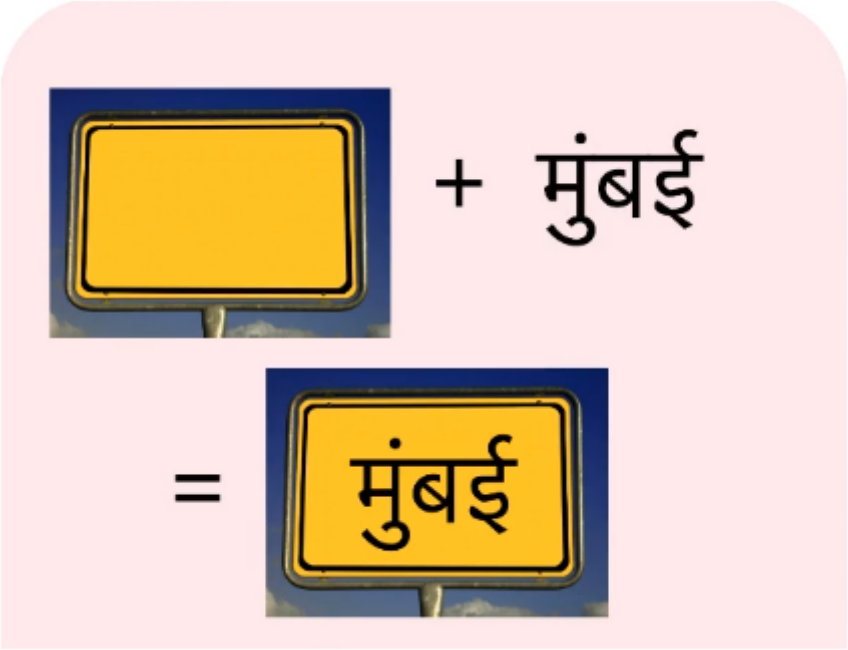
Now the question is **where do I get the data from?**

Data Curation: We have to curate this data.

There are lots of open data set repository available online:

There are some online platforms where we curate the data where for example we could upload a piece of an image and we can ask workers all around the world and they label/draw out the bounding box where the word appears.

create/get the empty signboard(using image editing tool we could create various templates of signboards), then we could take various names in Hindi, and then we use an image editing tool which takes these two things and pastes one on top of another. so, the net effect is that we get a lot of this signboard data.



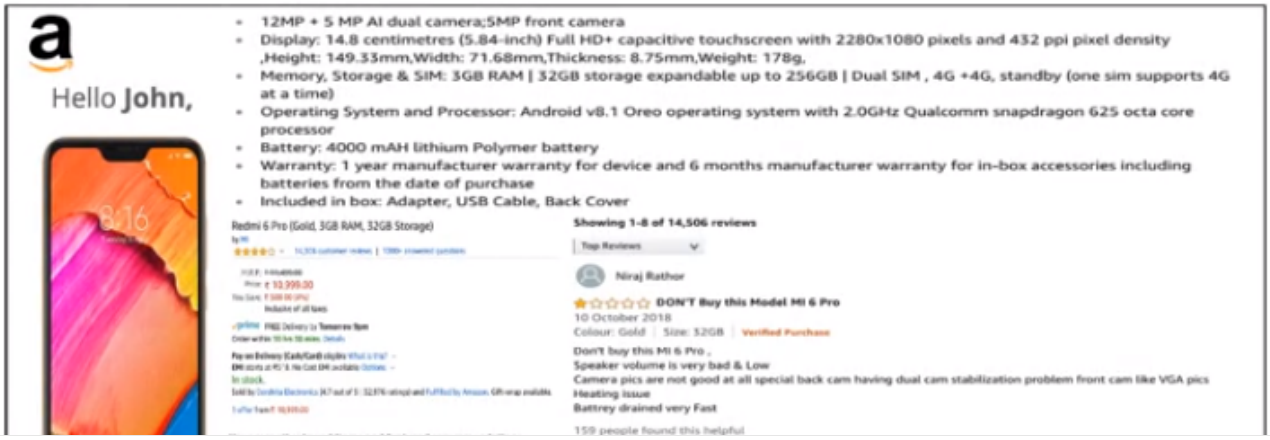
We could also get a lot of data on Wikidata.

Task:

A task is nothing but a crisp definition of what your input is and what your output is.

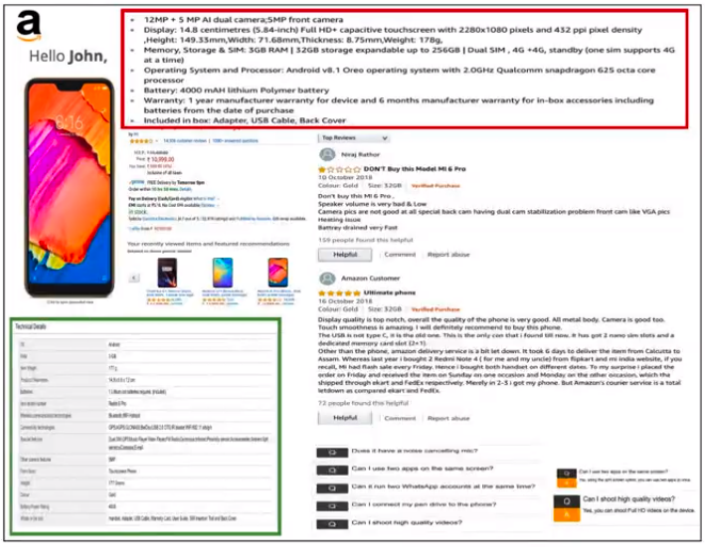
Once we have the data, what do we do with the data?

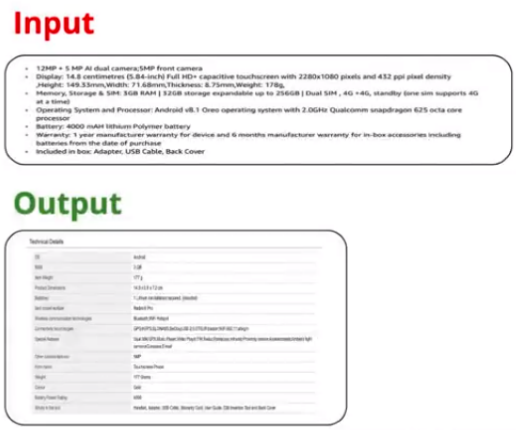
Below is a typical example of the data. We have various things over here, it is a product page from Amazon. Let's look at all the data that we have here.



We have image data, we have data in the structured description of the product, then we also have unstructured data as the reviews about the product, then we have the product description given by the Vendor which is a bit unstructured but also may have structure. Then we have the information about the related products which are similar to this product. Then we FAQs and other questions asked by users and their answers.

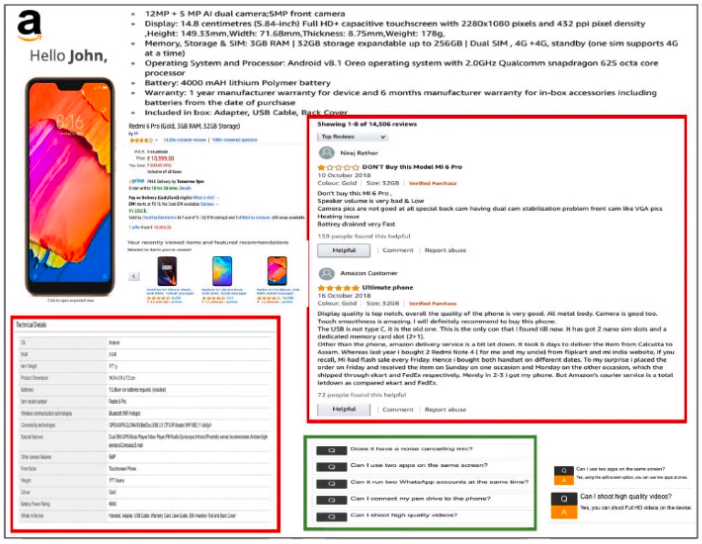
data is thrown at you what are the different tasks that you can do with this data.
For the above case, we could have many different tasks. One task could be:

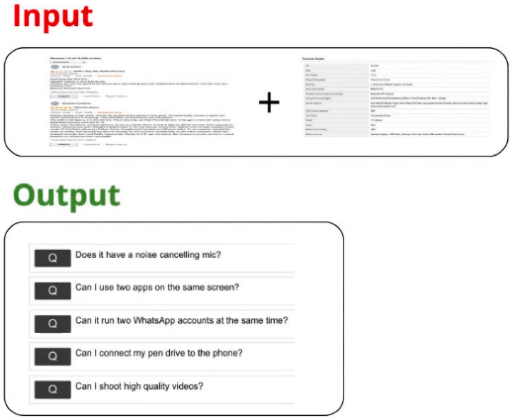




Given the data in the red box in the above image as the input which is some unstructured information about the product provided by the vendor, can we train a machine to fill in a table(green box in the above image) which would be a structured data? So, this is one of the tasks which could be accomplished using an abundant of the above data cases.

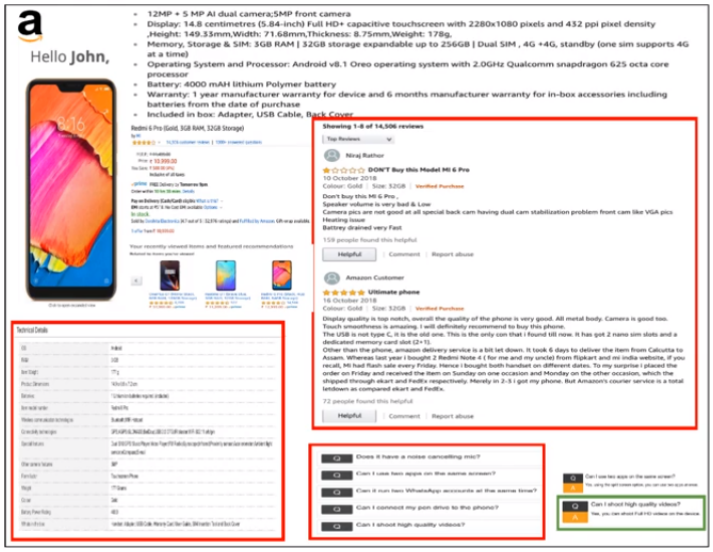
Another task could be:

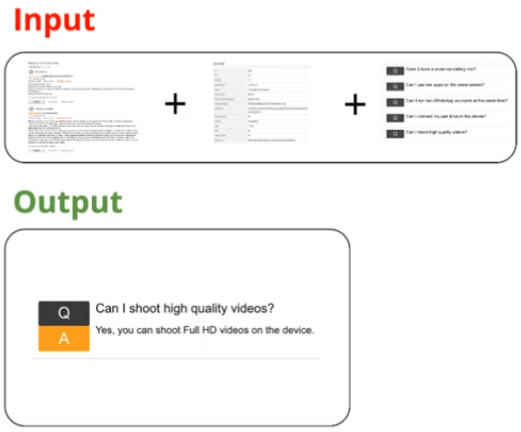




combination of these two), and using this we also have information about the FAQs that are important for this product. So, now if we are provided with new product reviews and specifications, the task is to come up with FAQs of this product.

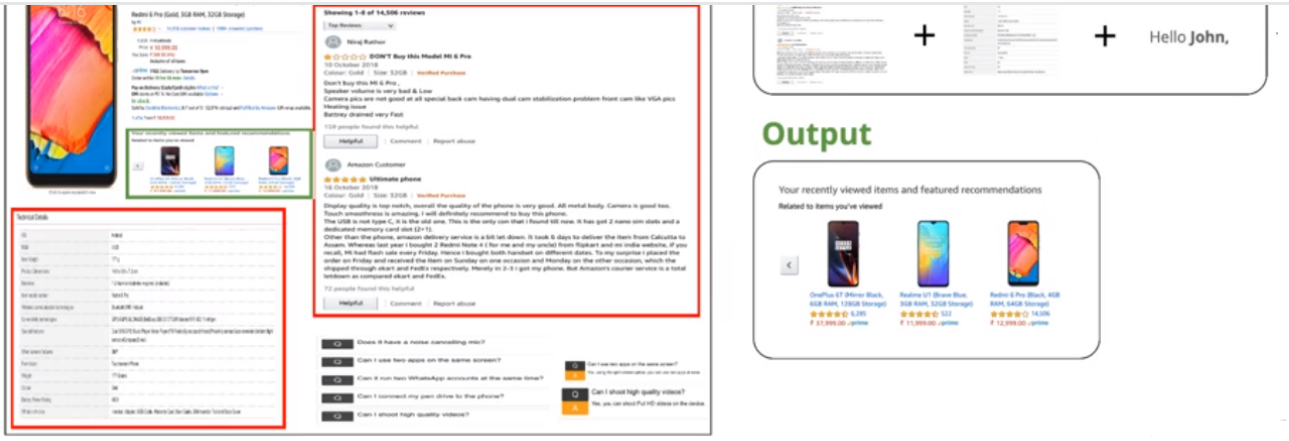
Yet another task we could have is:





Given the information about the reviews, product specification, and FAQs, can we answer any question that a random user is asking?

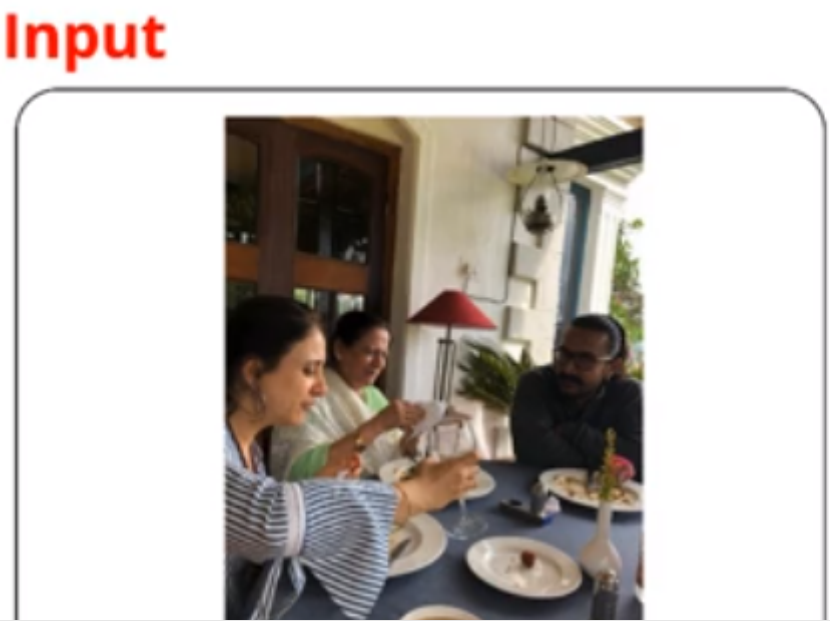
Another task that we could do here is that we have information about the user, we know what product page the user is looking at currently, what are the different attributes of that product and based on that we could recommend new products to this user based on his purchase history also.

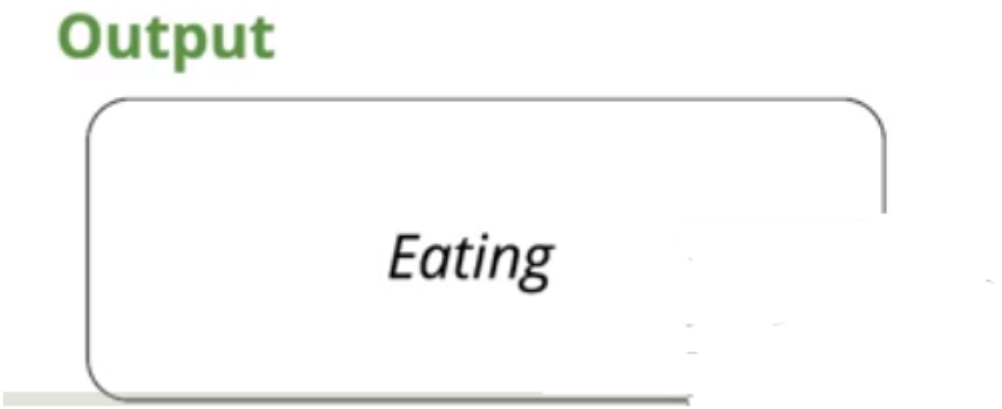


Let's take another example:

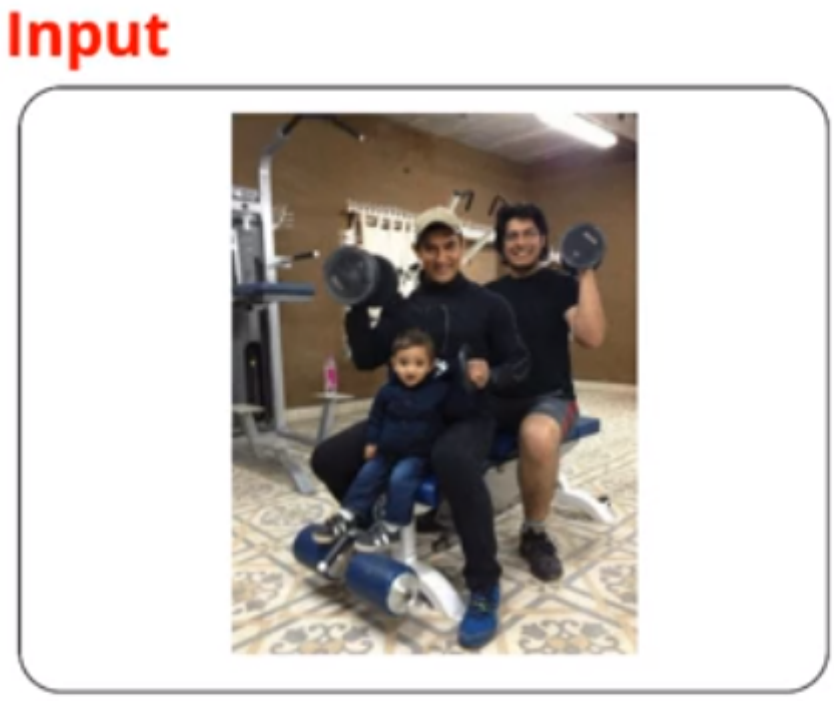
Here again, we have been provided with a lot of data. Several different tasks could be defined over here as well like:

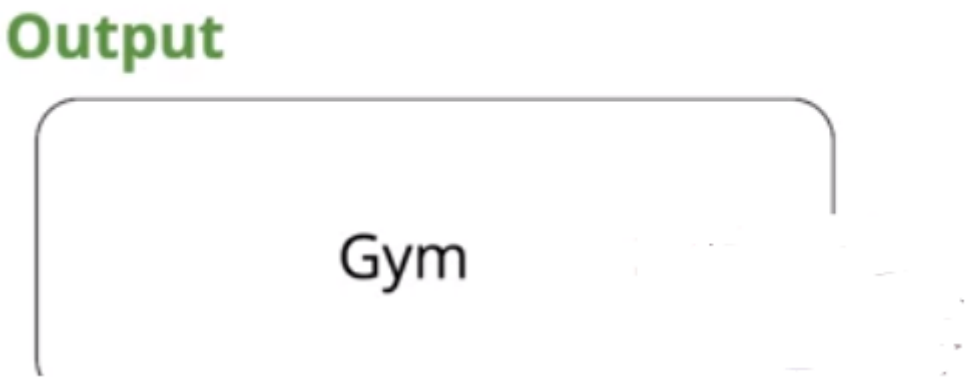
To identify activities from an image:





To identify location/place from a given image:



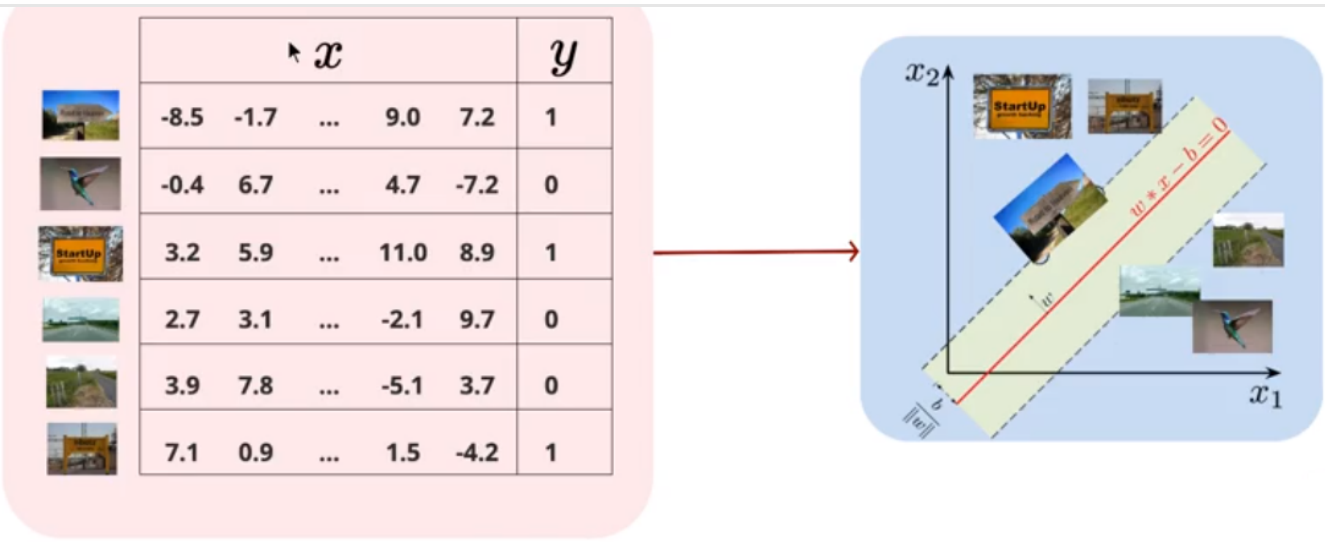


Another task would be: based on a facebook post information, we could recommend similar posts or similar activities.

Supervised

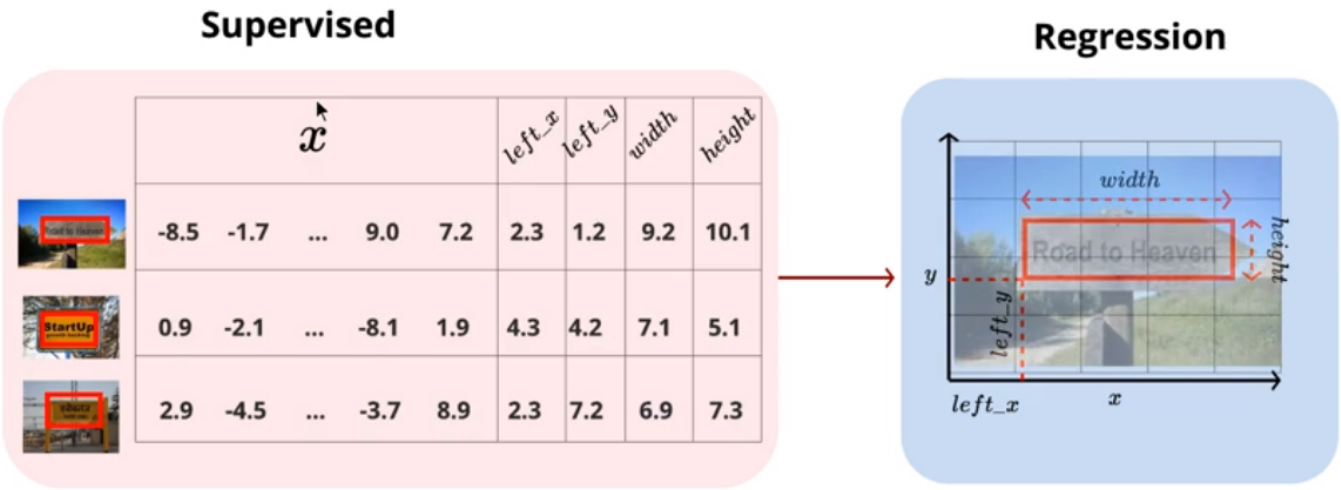
	x					y
	-8.5	-1.7	...	9.0	7.2	1
	-0.4	6.7	...	4.7	-7.2	0
	3.2	5.9	...	11.0	8.9	1
	2.7	3.1	...	-2.1	9.7	0
	3.9	7.8	...	-5.1	3.7	0
	7.1	0.9	...	1.5	-4.2	1

We can build a classifier that learns the relationship between x and y and the output of the classifier is 0 or 1 depending on whether the given input image contains text or it does not contain text.



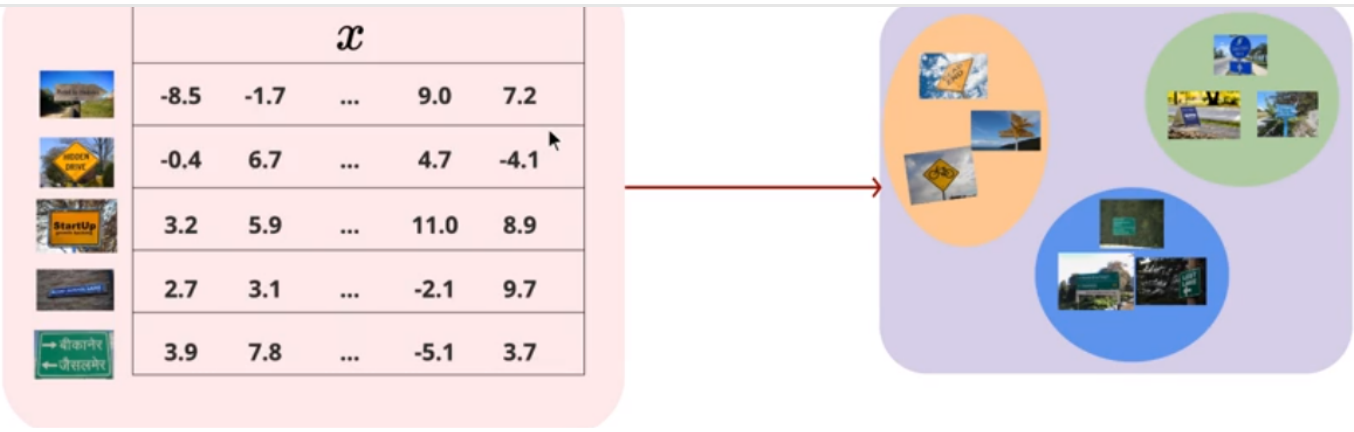
	x				y
	-8.5	-1.7	...	9.0	7.2	1
	-0.4	6.7	...	4.7	-7.2	0
	3.2	5.9	...	11.0	8.9	1
	2.7	3.1	...	-2.1	9.7	0
	3.9	7.8	...	-5.1	3.7	0
	7.1	0.9	...	1.5	-4.2	1

Another category of the task is Regression:

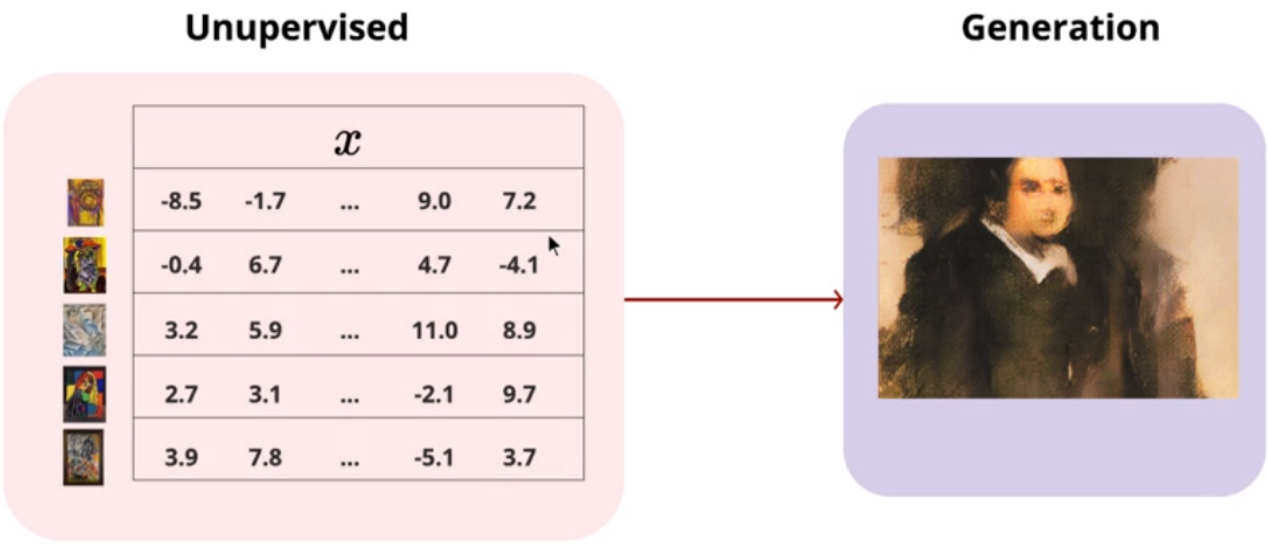


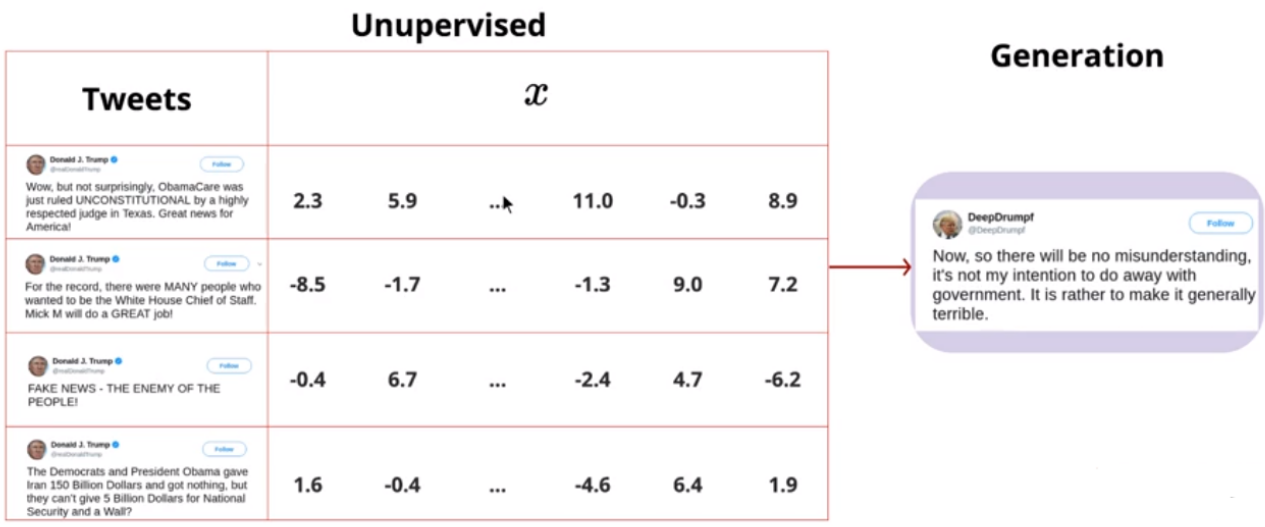
	x				*left_x*	*left_y*	*width*	*height*
	-8.5	-1.7	...	9.0	7.2	2.3	1.2	9.2	10.1
	0.9	-2.1	...	-8.1	1.9	4.3	4.2	7.1	5.1
	2.9	-4.5	...	-3.7	8.9	2.3	7.2	6.9	7.3

Another category of tasks comes under Unsupervised Tasks and one particular task here is clustering:



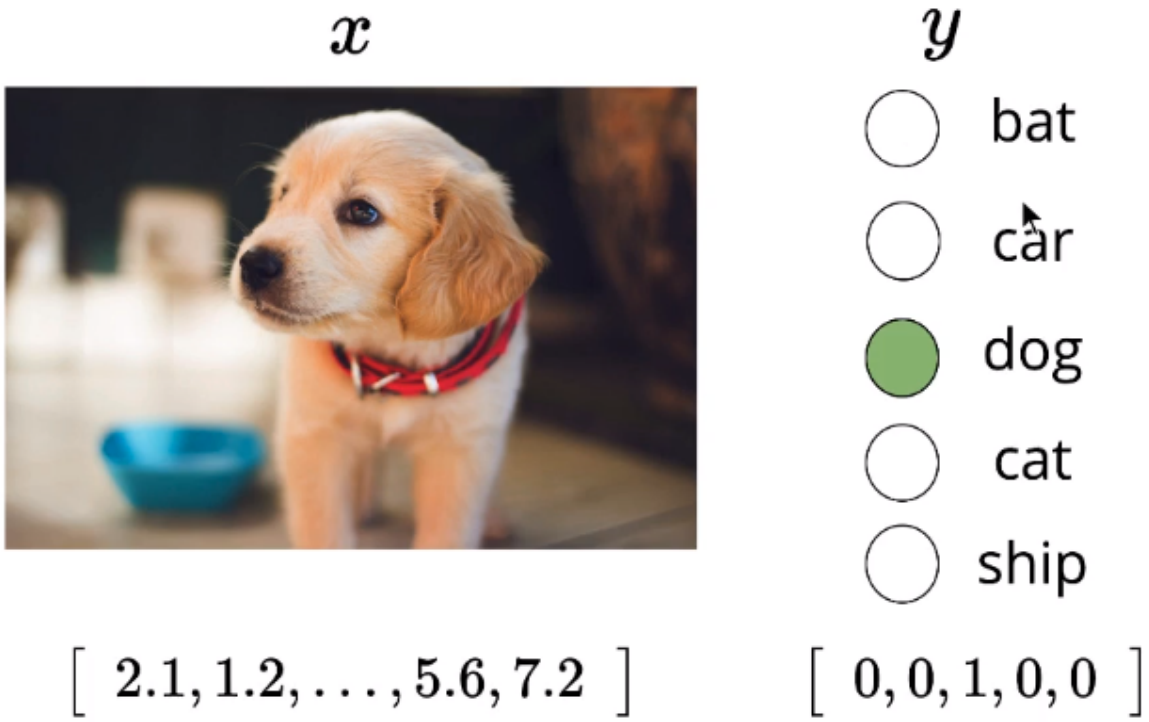
Another task is Generation:



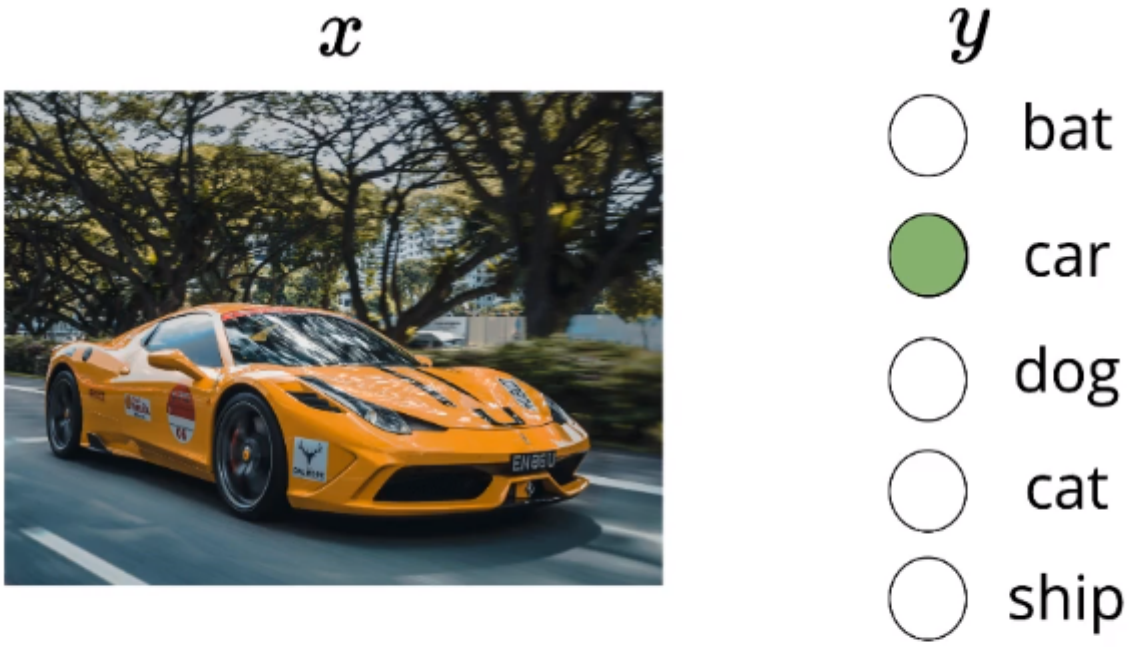


Model:

So far we have discussed the Data and Task jar.



Now we know that there is some true relationship that exists between the input and the output. The problem is that in most cases we don't know what this function is. What we typically do in machine learning is that we look at a lot of such data where we have **x** and **y** and in all of these cases we realize that there is some relation between x and y



x

y

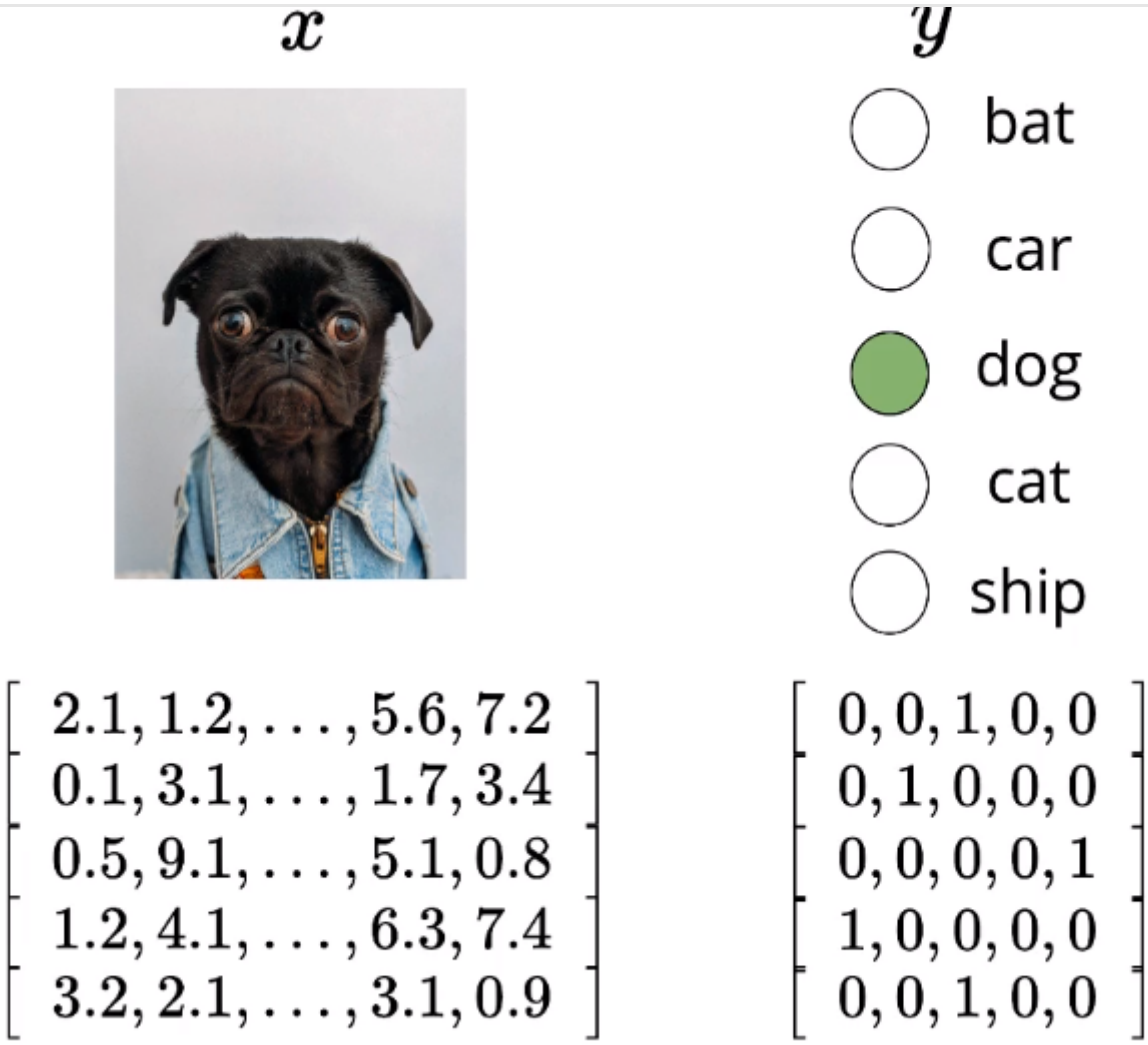
- bat
- car
- dog
- cat
- ship

$$\begin{bmatrix} 2.1, 1.2, \ldots, 5.6, 7.2 \\ 0.1, 3.1, \ldots, 1.7, 3.4 \\ 0.5, 9.1, \ldots, 5.1, 0.8 \\ 1.2, 4.1, \ldots, 6.3, 7.4 \end{bmatrix} \quad \begin{bmatrix} 0, 0, 1, 0, 0 \\ 0, 1, 0, 0, 0 \\ 0, 0, 0, 0, 1 \\ 1, 0, 0, 0, 0 \end{bmatrix}$$

$$y = f(x) \text{ [true relation, unknown]}$$

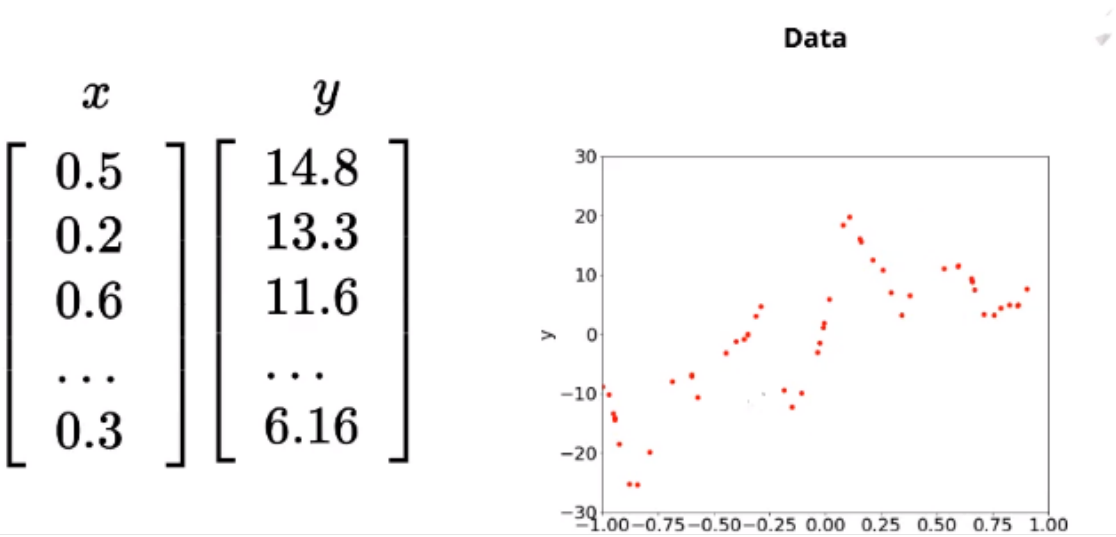
Now what we do is, we say that we don't know what the True relation is(between the input and the output), we come up with some function which we believe best approximates the relation between x and y

$$\hat{y} = \hat{f}(x) \text{ [our approximation]}$$



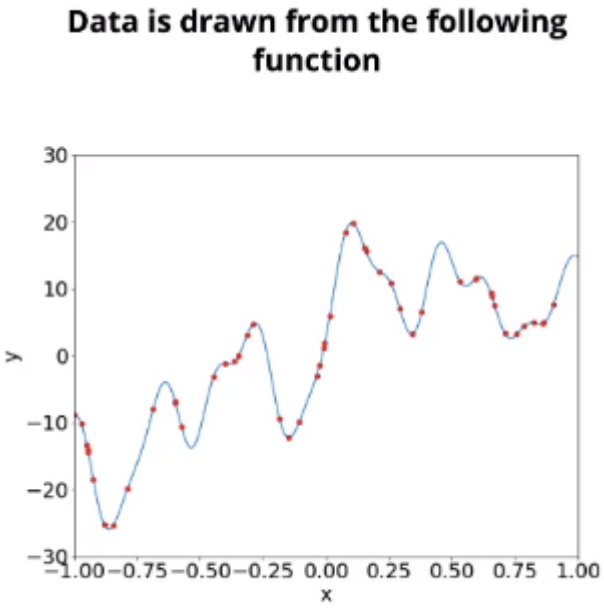
Let's understand this in much more detail.

We take one-dimensional data and plot it out



True relation between x and y is:

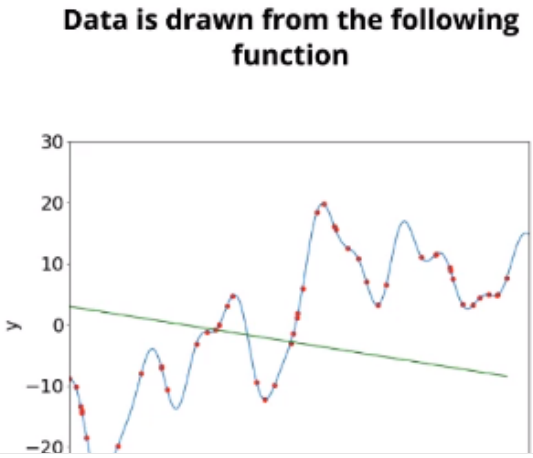
$$\begin{matrix} x \\ \begin{bmatrix} 0.5 \\ 0.2 \\ 0.6 \\ \dots \\ 0.3 \end{bmatrix} \end{matrix} \quad \begin{matrix} y \\ \begin{bmatrix} 14.8 \\ 13.3 \\ 11.6 \\ \dots \\ 6.16 \end{bmatrix} \end{matrix}$$



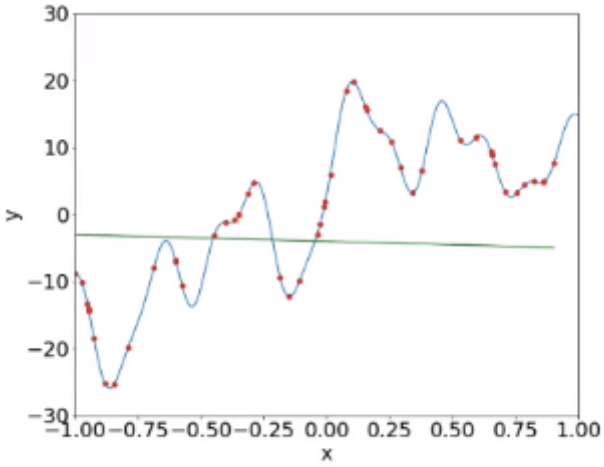
But we do not know the true function and will only have the data and using the data we need to come up with a function. So, we start with a very simple function say a linear function.

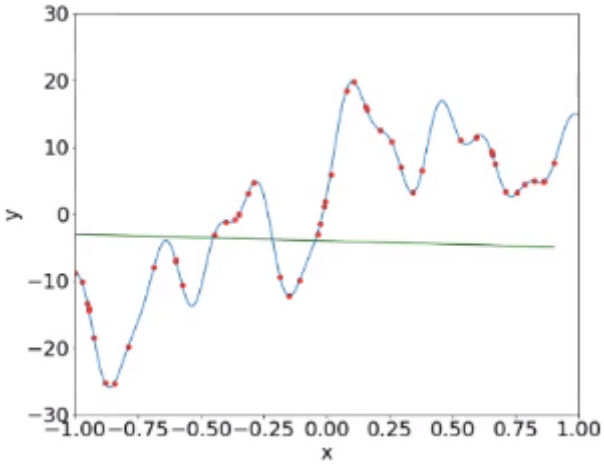
$$\hat{y} = mx + c$$

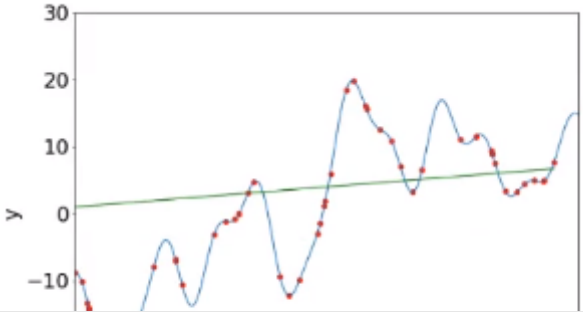
$$\begin{matrix} x \\ \begin{bmatrix} 0.5 \\ 0.2 \\ 0.6 \\ \dots \\ 0.3 \end{bmatrix} \end{matrix} \quad \begin{matrix} y \\ \begin{bmatrix} 14.8 \\ 13.3 \\ 11.6 \\ \dots \\ 6.16 \end{bmatrix} \end{matrix}$$

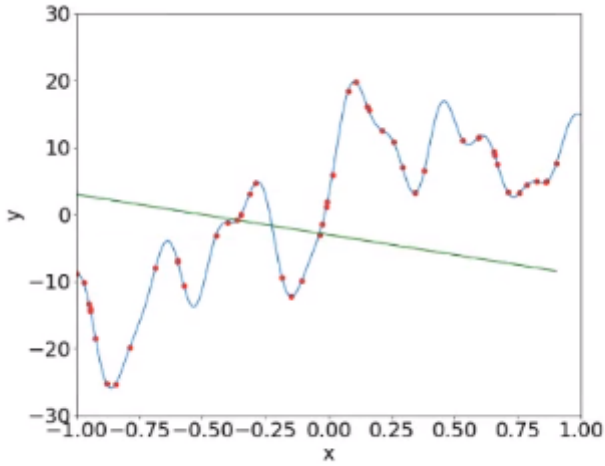


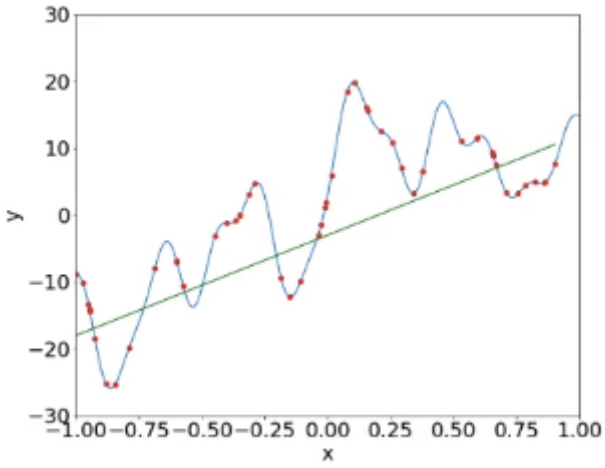
The parameters m and c are learned using all the data that we have. We can learn these two parameters even if we have just 2 data points but the problem here is that no matter how we adjust these parameters m and c, we can not come up with a line that passes through all of these points.

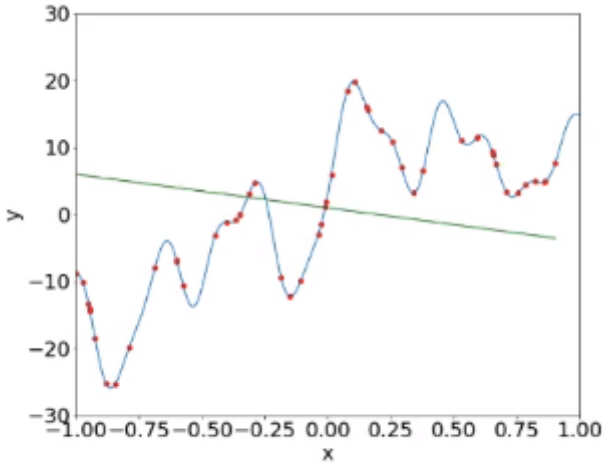






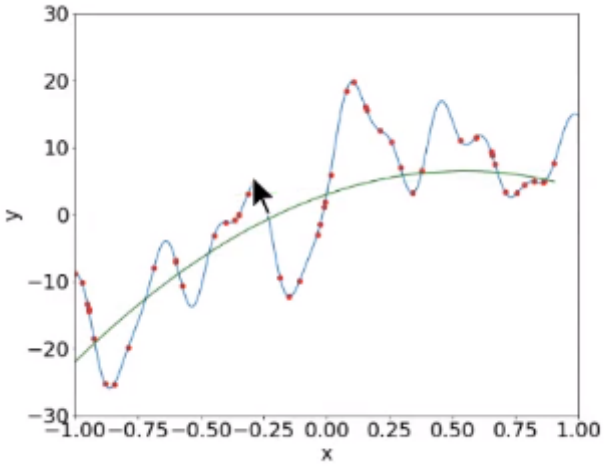

30
20
10
0
−10
−20
−30
y
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

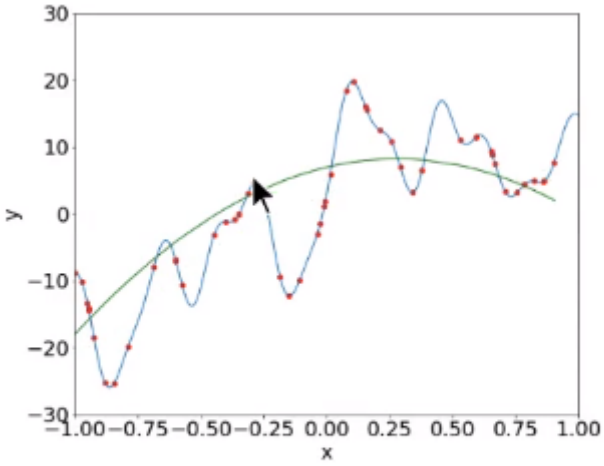

30
20
10
0
−10
−20
−30
y
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

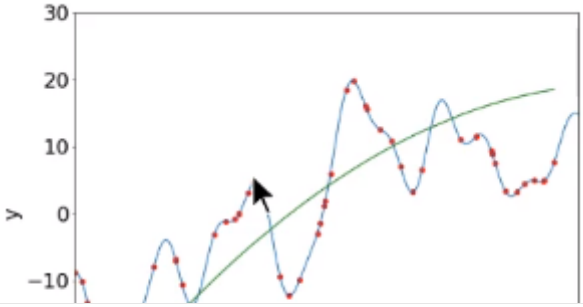

30
20
10
0
−10
−20
−30
y
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

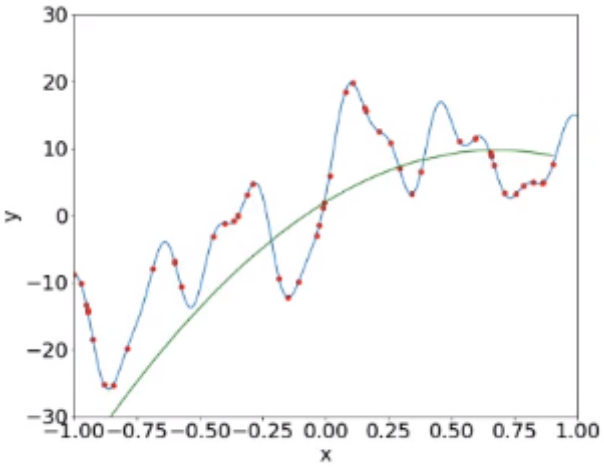
data.

$$\hat{y} = ax^2 + bx + c$$







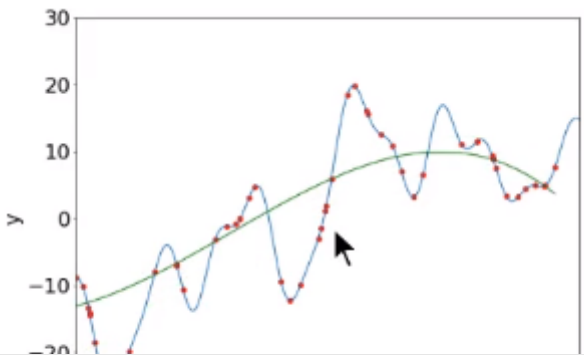


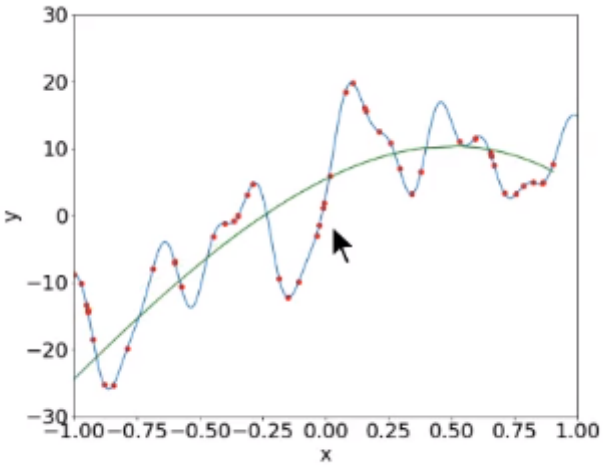
Here also we see that no matter what different values of a, b and c we try, we are not able to get a situation where for any given **x** here the output of the function(green curve) is the same as the red point(corresponding to the same x). So, this function is also not complex enough to capture the relationship between **x** and **y**.

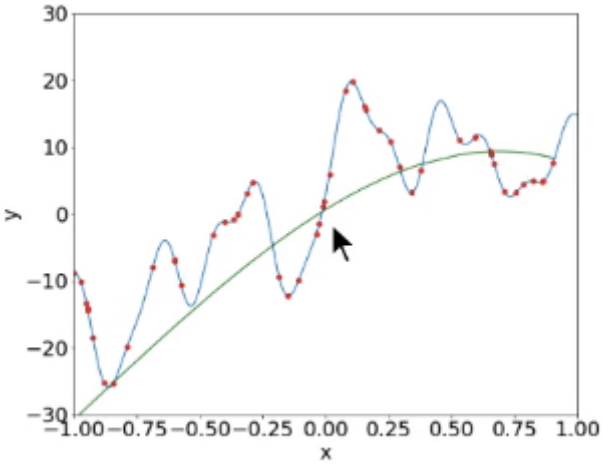
The machine is trying to find the values of the parameters very efficiently in such a way that the value of error is minimum that means we should be as close to the predictions as possible.

So, we try again with degree 3 polynomial.

$$\hat{y} = ax^3 + bx^2 + cx + d$$



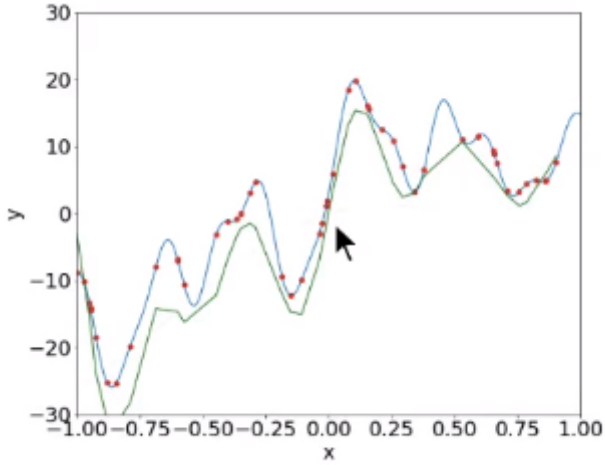


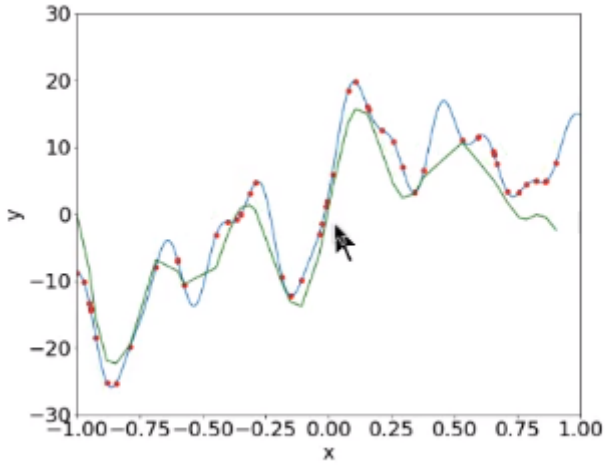


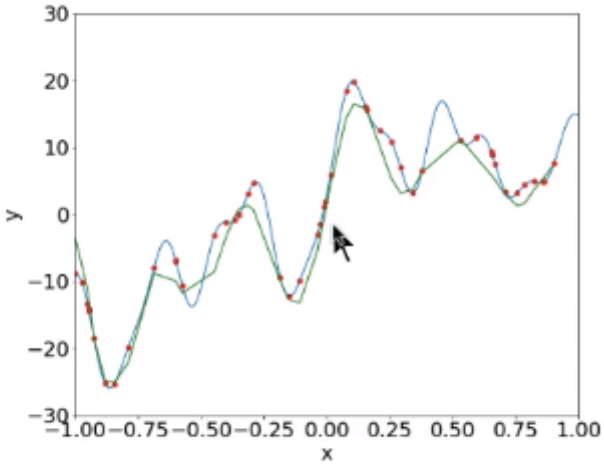
And once again the same story repeats, we try to adjust the values of a, b, c and d such that the green curve is as close to the red dots as possible.

And we keep trying this with higher degree polynomials until we reach to degree 25 polynomial which is coming very close not exactly overlapping with the red points but still very close and captures the relationship between x and y.

$$\hat{y} = ax^4 + bx^3 + cx + d$$
$$\vdots$$


30
20
10
0
−10
−20
−30
y
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x


30
20
10
0
−10
−20
−30
y
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

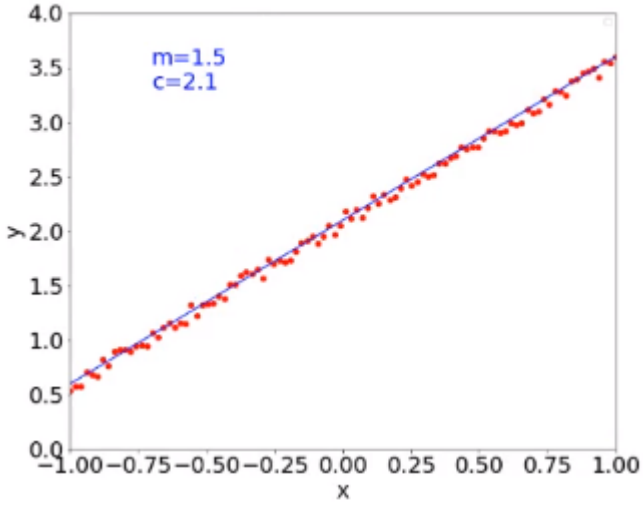

30
20
10
0
−10
−20
−30
y
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x

And in most of the real-world problems, data would be a high dimensional meaning we could not just plot this out and visualize which function would be the best fit for this data set.

One question that arises now is that why not try a very complex model from the very beginning?

The reason for this is that suppose the true relation between x and y is very close to a line.

$$\begin{matrix} x \\ \begin{bmatrix} 0.1 \\ 0.2 \\ 0.4 \\ .... \\ 0.8 \end{bmatrix} \end{matrix} \quad \begin{matrix} y \\ \begin{bmatrix} 2.6 \\ 2.4 \\ 3.1 \\ .... \\ 4.1 \end{bmatrix} \end{matrix}$$

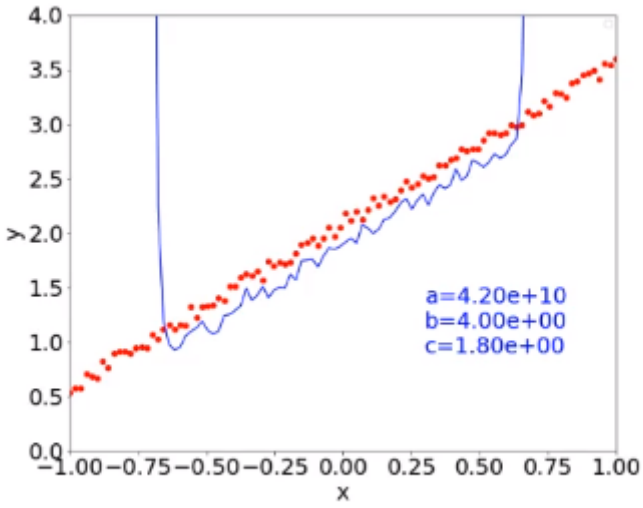


$$y = mx + c \text{ [true function, simple]}$$

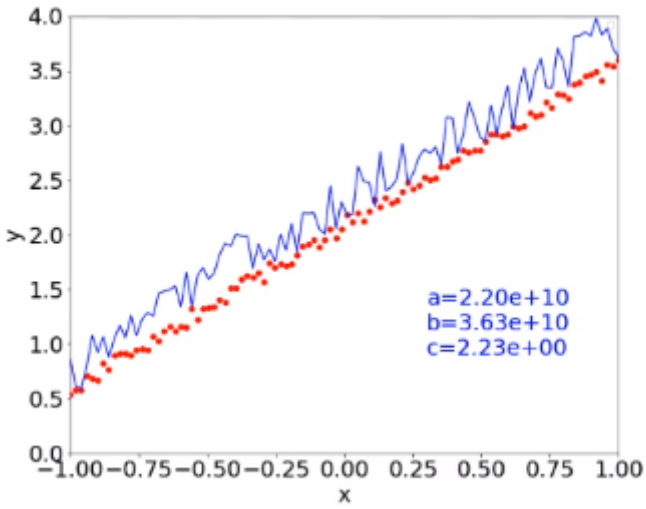
And what if we try it to approximate using a very complex function which is say a 100-degree polynomial

$$\hat{y} = ax^{100} + bx^{99} + \ldots + mx + c$$

[our approximation, very complex]



In this case, we could say that the machine should be able to find all the parameters such that all the parameters expect m and c are zero, that means we could still start with a very complex function and the machine could learn to ignore it and come up with a very simple function. But if we try this, then we would get stuck at a point where it would be very difficult for the machine to learn all these 98 parameters to be 0. Although it seems easy for a human being, the machine has to try out all these values and out of these infinite values, it has to find out this exact peculiar case where all the parameters are 0 except m and c. So, in practice, it would be very difficult for the machine to do that.

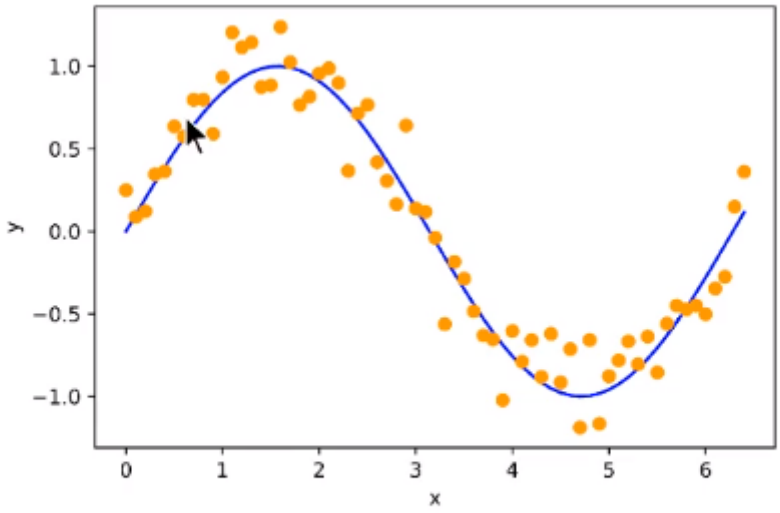


Loss Function:

How do we know which model is better?

	x	y
1	0.00	0.24
2	0.10	0.08
3	0.20	0.12
....	....	....
n	6.40	0.36

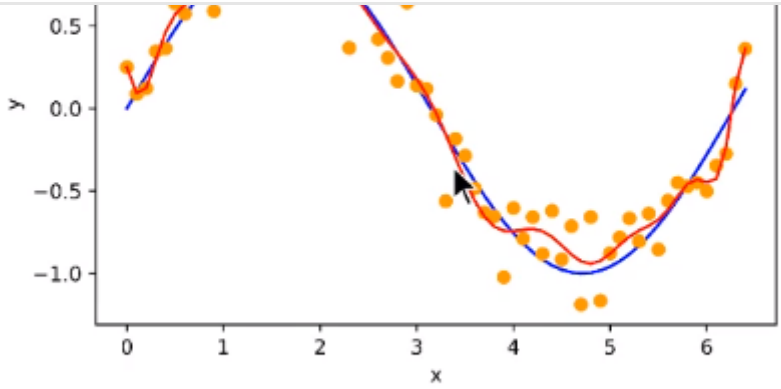
And the following is the **true relation** between x and y which **is something very close to the sine function along with some noise**.



True Function

Someone approximates this function as

$$\hat{f}_1(x) = 1.79x^{25} - 4.54x^{24} + \ldots - 1.48x + 2.48$$

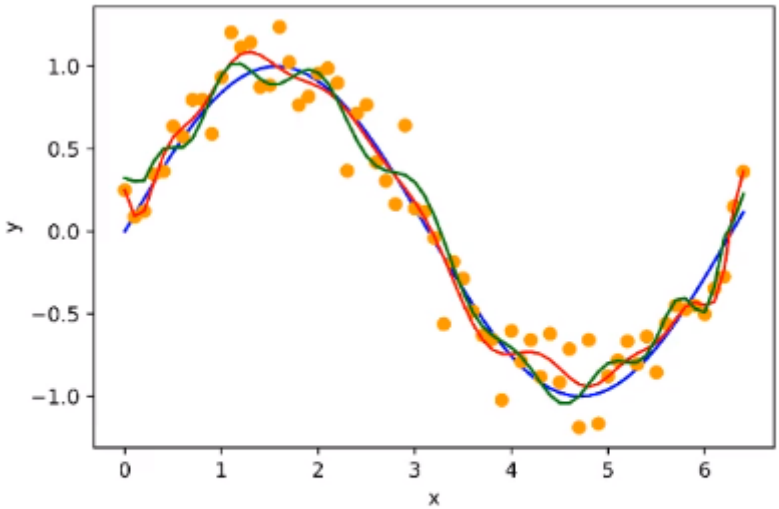


$$\hat{f}_1(x)$$

Red curve

Now another person has come up with let's say the below function:

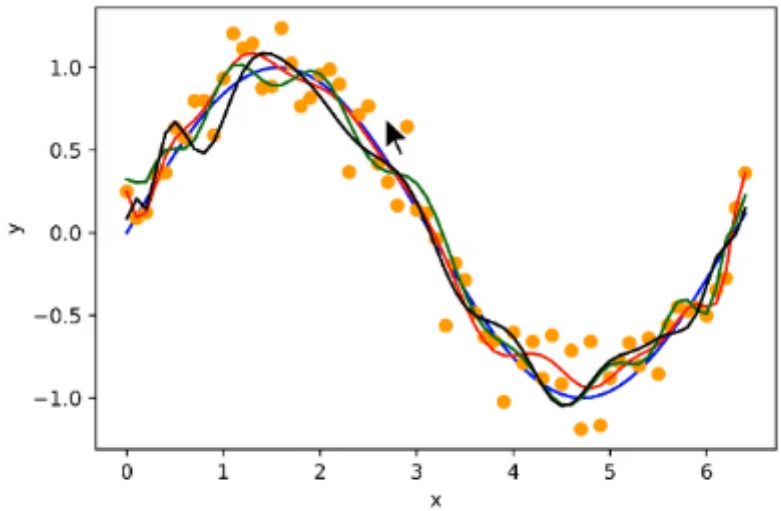
$$\hat{f}_2(x) = 2.27x^{25} + 9.89x^{24} + ... + 2.79x + 3.22$$



$$\hat{f}_2(x)$$

Green curve

Now the third person came up with the following function:



$\hat{f}_3(x)$

Black Curve

All these 3 functions seem to be very close to the sine function. Now the question is which of these functions is better?

If we try to visually inspect, it's not very clear as all the curves are very close to each other and to the true curve. So, instead of looking at the curves, we could look at some numbers and decide which function is better.

	x	y	$\hat{f}_1(x)$	$\hat{f}_2(x)$	$\hat{f}_3(x)$
1	0.00	0.24	0.25	0.32	0.08
2	0.10	0.08	0.09	0.30	0.20
3	0.20	0.12	0.11	0.31	0.14
....	....	....	....	....	....
n	6.40	0.36	0.36	0.22	0.15

+ve value of error does not cancel out with the -ve value of error and there is also a calculus-based reason for this) of the difference between the true output and the predicted values for all the data points.

$$\mathscr{L}_1 = \sum_{i=1}^{n} (y_i - \hat{f}_1(x_i))^2$$
$$\mathscr{L}_2 = \sum_{i=1}^{n} (y_i - \hat{f}_2(x_i))^2$$

$$\begin{aligned}\mathscr{L}_1 &= \sum_{i=1}^{n} (y_i - \hat{f}_1(x_i))^2 \\ &= (0.24 - 0.25)^2 + (0.08 - 0.09)^2 + (0.12 - 0.11)^2 + \\ &\qquad \ldots + (0.36 - 0.36)^2 \\ &= 1.38\end{aligned}$$

In a similar way, we could compute this error value for all the three functions that we approximated.

$$\mathscr{L}_1 = \sum_{i=1}^{n} (y_i - \hat{f}_1(x_i))^2 = 1.38$$

$$\mathscr{L}_2 = \sum_{i=1}^{n} (y_i - \hat{f}_2(x_i))^2 = 2.02$$

$$\mathscr{L}_3 = \sum_{i=1}^{n} (y_i - \hat{f}_3(x_i))^2 = 2.34$$

And now based on this error value we could say that the first approximation was better than the other two. **So, a loss function helps us decide how good or bad our model is or how good or bad our parameters are. And it also helps to decide the better model among a given set of models.**

Learning Algorithm:

We have been provided with the Data and the task as well:

Budget in (100crs)	Box Office Collection in (100 crs)	Action Scene in times (100 mins)	IMDB Rating
0.55	0.66	0.22	4.8
0.68	0.91	0.77	7.2
0.66	0.88	0.67	6.7
:	:	:	:
0.72	0.94	0.97	8.1
0.58	0.74	0.35	5.3

We have also proposed a model as the following

$$\hat{f}(x) = ax_1^2 + bx_2^3 + cx_3^2$$

Let's say somehow we also got the parameters of this model as:

$$\hat{f}(x) = 3.5x_1^2 + 2.5x_2^3 + 1.2x_3^2$$

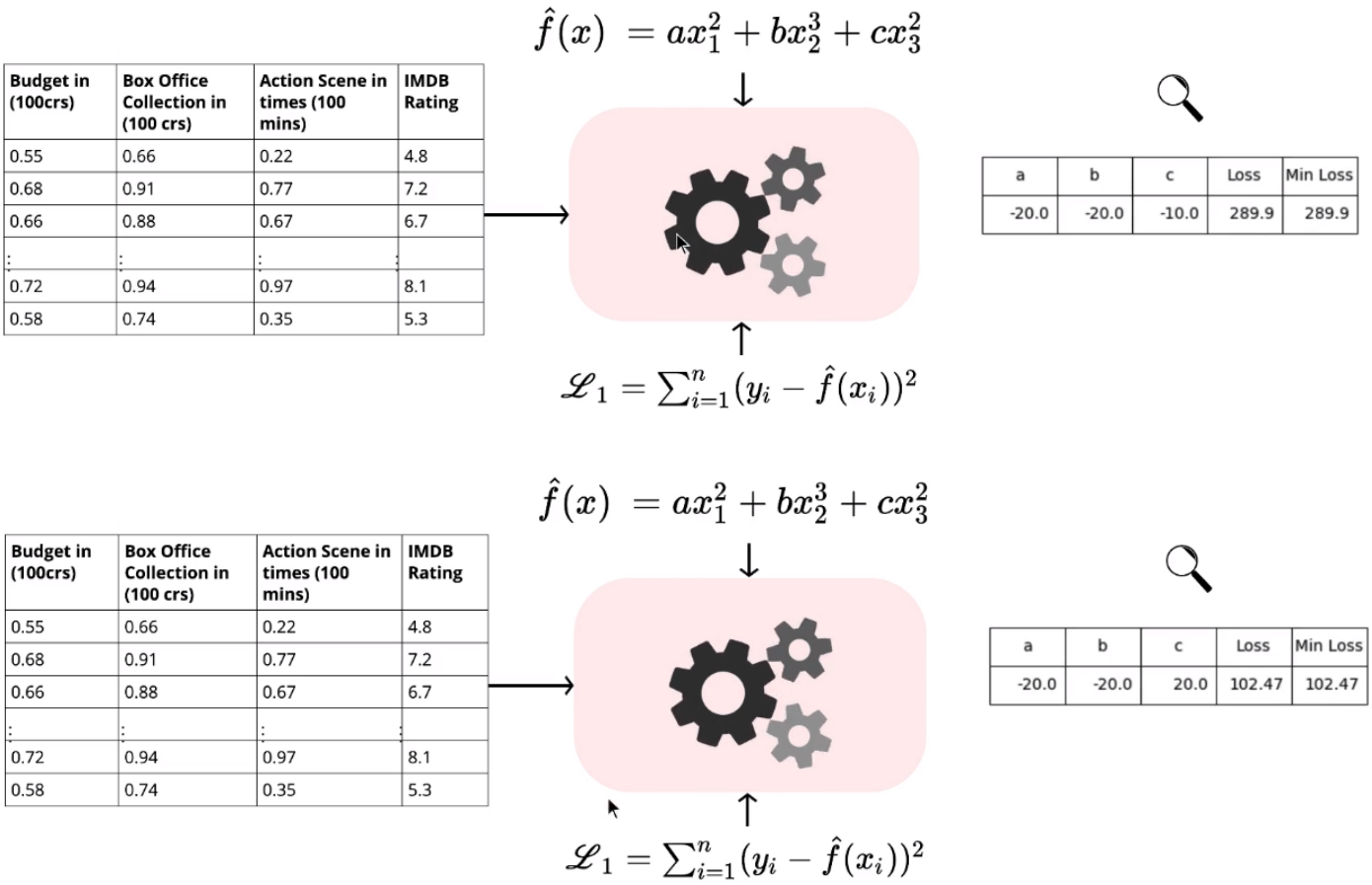
Then we can check the error value of this model as using the below:

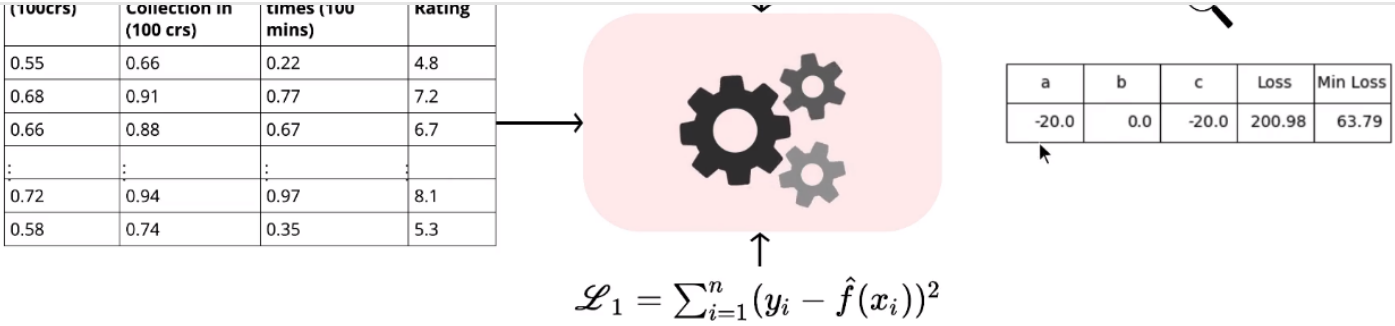
$$\mathscr{L}_1 = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

Till now we have the Data, Task, Model, and Loss function. We don't have a way to learn the parameters of the model.

Learning Algorithm helps us to learn the parameter of the model.

We could then fix the value of the **a and b both to -20 and then search across c(try out different value from -20 to 20) and plug these values into the Loss function(plug into model equation, then compute the predicted output and from there we compute the loss) and see what value of c gives the best result.** And then we repeat the process and do the same thing and find the value of b for which the loss is minimum. In other words, in this entire space of -20 to 20, for all values of a, b and c, we compute the loss value by plugging in these values of a, b and c in the model/function and using in the Loss function equation. We also keep track of the minimum loss.

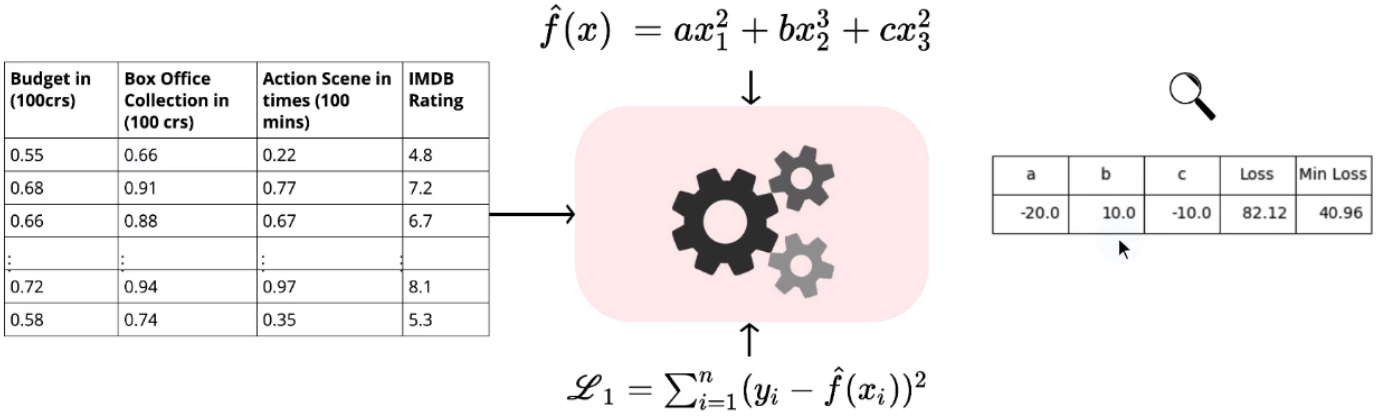




(100crs)	Collection in (100 crs)	times (100 mins)	Rating
0.55	0.66	0.22	4.8
0.68	0.91	0.77	7.2
0.66	0.88	0.67	6.7
⋮	⋮	⋮	⋮
0.72	0.94	0.97	8.1
0.58	0.74	0.35	5.3

a	b	c	Loss	Min Loss
-20.0	0.0	-20.0	200.98	63.79

$$\mathscr{L}_1 = \sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

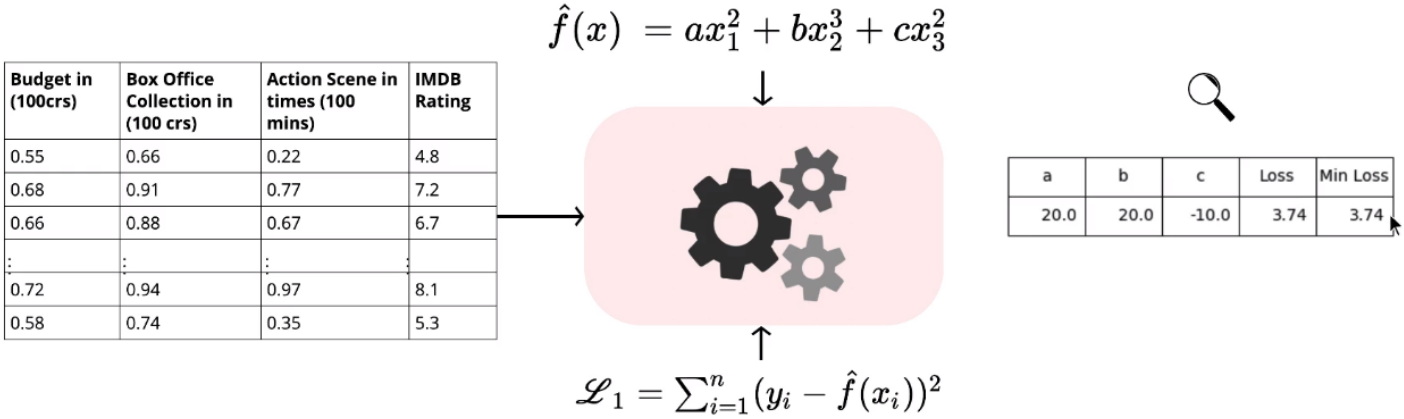


$$\hat{f}(x) = ax_1^2 + bx_2^3 + cx_3^2$$

Budget in (100crs)	Box Office Collection in (100 crs)	Action Scene in times (100 mins)	IMDB Rating
0.55	0.66	0.22	4.8
0.68	0.91	0.77	7.2
0.66	0.88	0.67	6.7
⋮	⋮	⋮	⋮
0.72	0.94	0.97	8.1
0.58	0.74	0.35	5.3

a	b	c	Loss	Min Loss
-20.0	10.0	-10.0	82.12	40.96

$$\mathscr{L}_1 = \sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$



$$\hat{f}(x) = ax_1^2 + bx_2^3 + cx_3^2$$

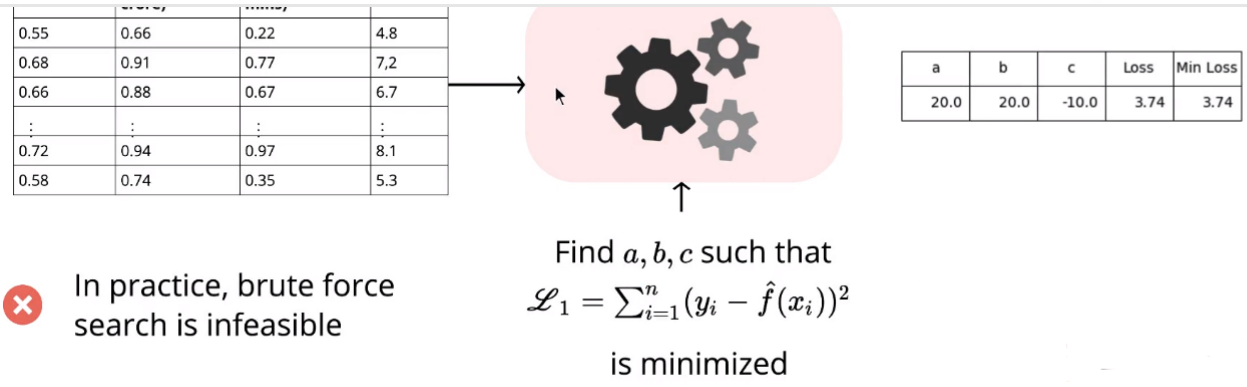
Budget in (100crs)	Box Office Collection in (100 crs)	Action Scene in times (100 mins)	IMDB Rating
0.55	0.66	0.22	4.8
0.68	0.91	0.77	7.2
0.66	0.88	0.67	6.7
⋮	⋮	⋮	⋮
0.72	0.94	0.97	8.1
0.58	0.74	0.35	5.3

a	b	c	Loss	Min Loss
20.0	20.0	-10.0	3.74	3.74

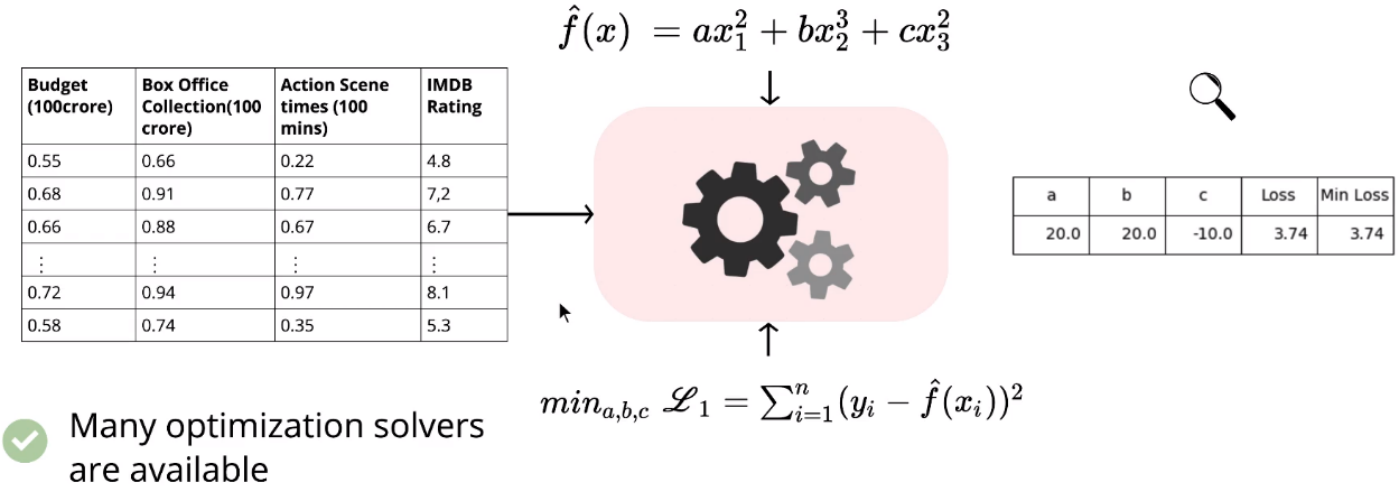
$$\mathscr{L}_1 = \sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

At the end of this search process, we will get some value of a, b and c for which the loss is minimized. And we will just output those values. And that would be the parameter configuration for which the loss is minimized.

So, this is a very brute force approach. In practice, this brute force approach would not be feasible because we would not only have just 3 parameters, we would have thousands of values. So, what we need is an efficient way of computing these parameters.



So, now this is converted to an optimization problem. **We need to find values of a, b and c for which the Loss value is minimized.** So, we have a function(Loss function) which depends on the values of a, b and c and we want to find the values of a, b, and c such that the value of that function is minimized.

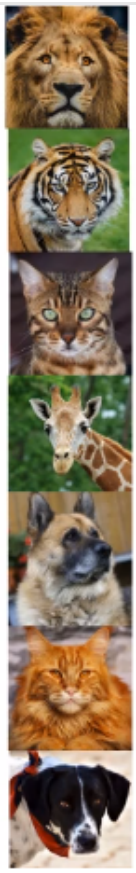


And the way we go about computing the parameters which give the minimum value of the Loss function lies in the Calculus where we take the partial derivative of the function with respect to the parameter(s) and equate this to 0 to get the value of the parameter(s) for which the function's value would be minimum.

Evaluation:

How do we evaluate our model?

Let's say we are building an image classifier. We pass the data through the model, predict the output and we also know the true values/labels



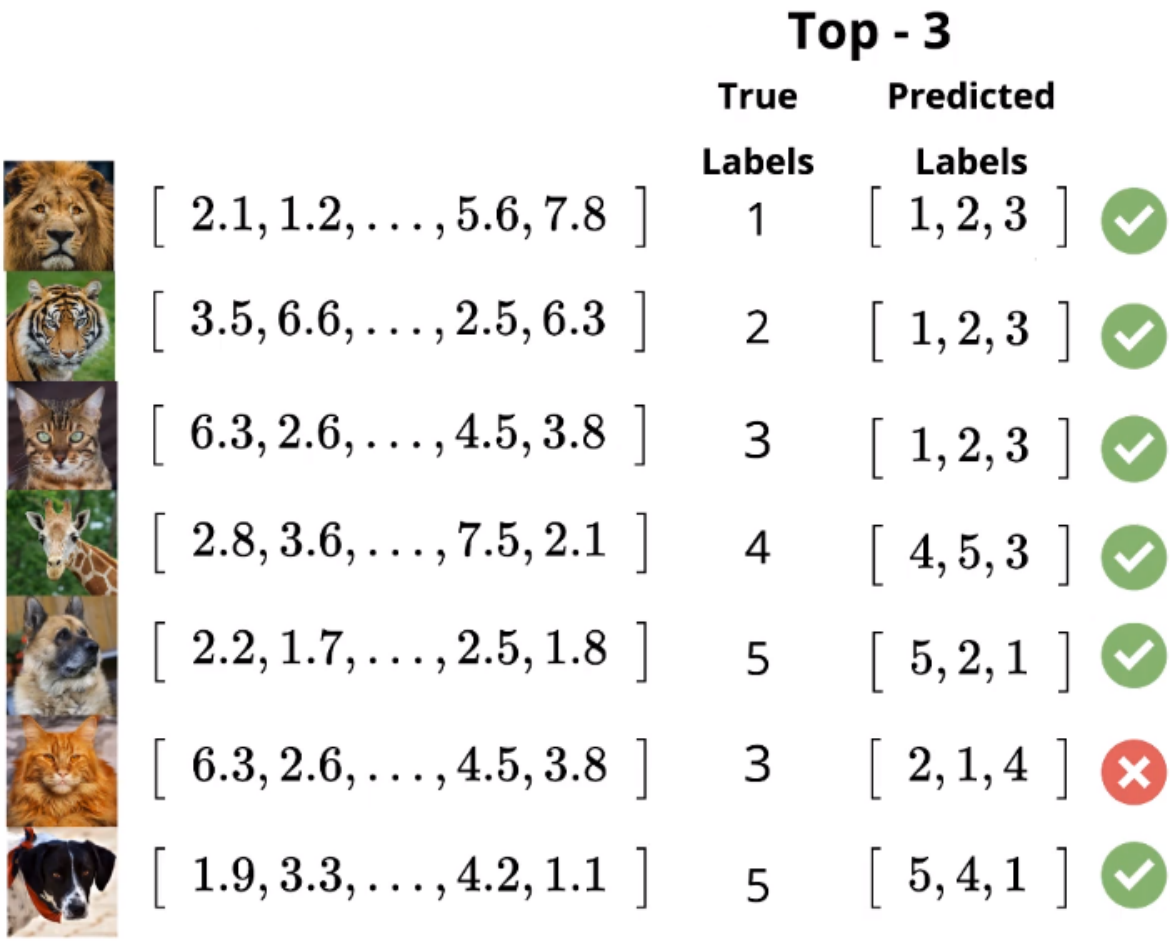
$[\ 2.1, 1.2, \ldots, 5.6, 7.8\ ]$	1	1	✓
$[\ 3.5, 6.6, \ldots, 2.5, 6.3\ ]$	2	3	✗
$[\ 6.3, 2.6, \ldots, 4.5, 3.8\ ]$	3	1	✗
$[\ 2.8, 3.6, \ldots, 7.5, 2.1\ ]$	4	4	✓
$[\ 2.2, 1.7, \ldots, 2.5, 1.8\ ]$	5	5	✓
$[\ 6.3, 2.6, \ldots, 4.5, 3.8\ ]$	3	2	✗
$[\ 1.9, 3.3, \ldots, 4.2, 1.1\ ]$	5	5	✓

Class Labels	
Lion	1
Tiger	2
Cat	3
Giraffe	4
Dog	5

And **we can just compute the accuracy of the model. The way we do this is that for each instance we compare the output predicted by the model with the true output.** So, in the above image, out of the total 7 cases provided, the model predicted the correct output in 4 cases. And the accuracy, in this case, would be:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

So, this is one way of evaluating the models. **In many practical situations, we might look at the top three outputs given by the model and we are okay as long as one of those three matches the true output.** And this is termed as **Top-3 accuracy or in general Top-k accuracy if we are looking at the top k outputs**.



$$Accuracy = \frac{\text{Number of correct predictions in top-3}}{\text{Total number of predictions}}$$

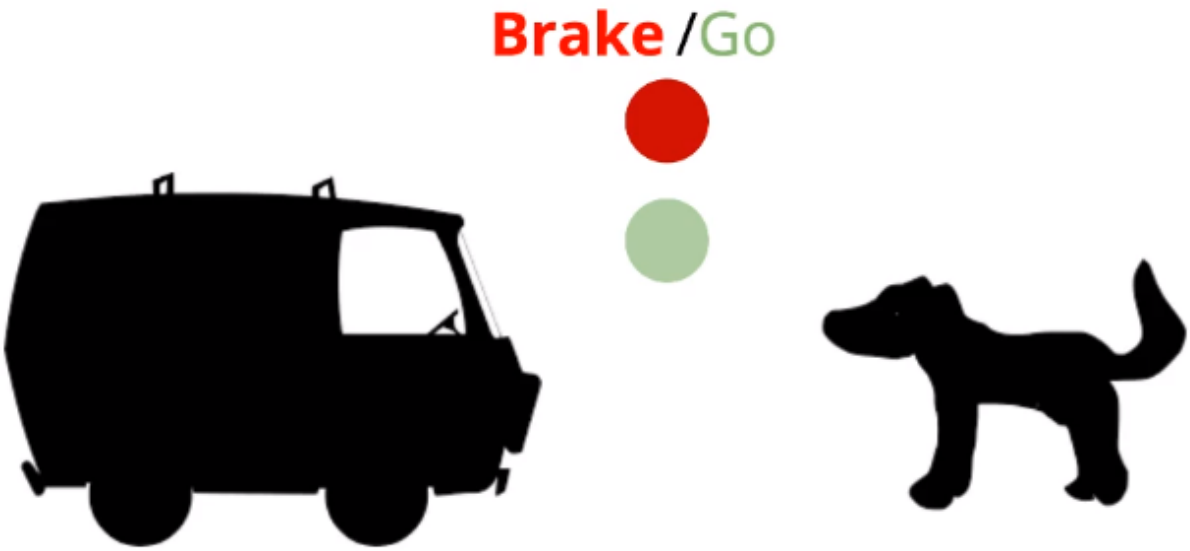
$$= \frac{6}{7} = 0.86$$

Now the question arises is that, how this is different from the Loss Function?

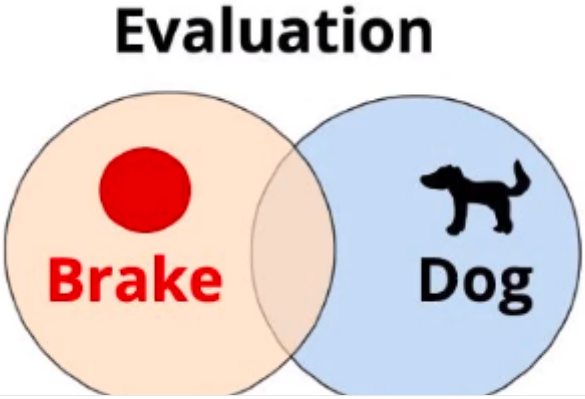
if we are told that the Square Error loss is 0.4, it's hard for us to judge how good or bad our model is.

And here is also some practical intuition why we should have a different measure for Loss function and different measures for evaluating the model:

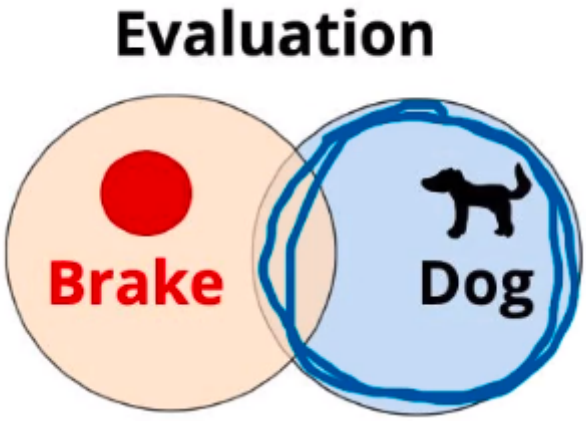
So, think of this practical application where we building an autonomous driving car and one module would be to deciding whether to apply the brake or not depending on if there is an obstacle(like a dog) or not.



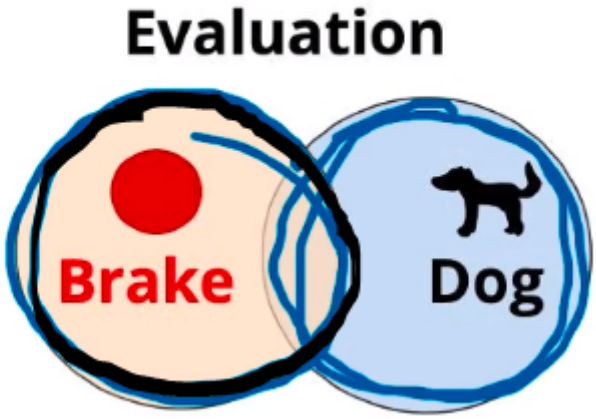
So, the model's job is to apply break whenever there is an obstacle and if there is no dog(let's say the dog is the only obstacle on the road), then the car should just keep going. The standard evaluation metric for this case is something known as precision and recall and we calculate/understand it using the Venn diagram.



In the below image, the highlighted(circled) portion indicates all the cases when there was an obstacle on the road.



And the highlighted portion(in black) in the below image represents all the cases when the model decided to apply the brake.



In a perfect world, we would like the two circles to be perfectly overlapping meaning every time there was a dog we applied the brake, and whenever we applied the brake there was some obstacle. We never applied the brake when there was no obstacle or dog.

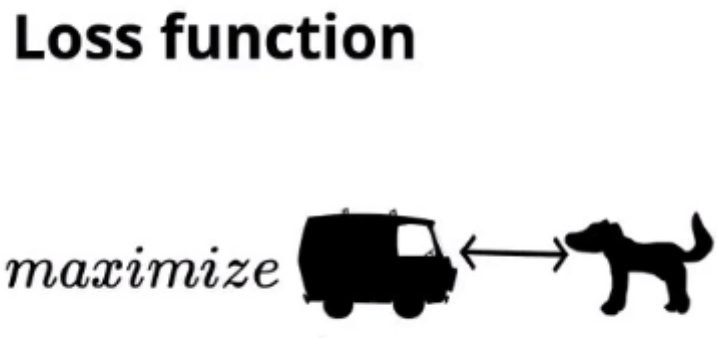
Brake circle represents the total number of times we applied the brake and the intersection region represents the number of times when we correctly applied the

And **recall in this case would be the total number of times there was a dog and we applied the brake(intersection part) divided by the total number of times the dog was there**. So, precision is very intuitive, that the number of times you took the correct action out of the total number of times you took an action, of all the times you took an action how many times were you correct.

And recall is of how many times you were supposed to take any action, how many times did you actually take an action.

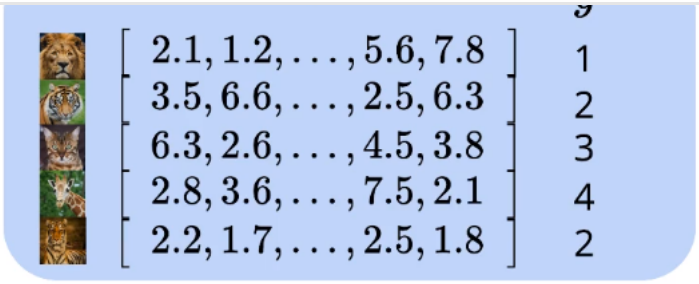
So, this Precision and recall we could as Evaluation metic.

But if we were to train this model, we could use a Loss function which tries to maximize the distance between the car and the obstacle(we don't want to apply the brake very close to the dog, there might be a possibility of hurting the dog or some unknown event there). So, that's how we would like to train our model to not just learn when to apply the brake but to also maximize the distance between the car and obstacle.

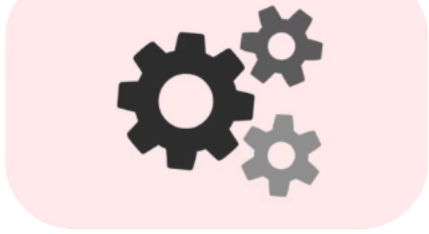


So, as is clear from this example the Loss function and Evaluation metrics are very different.

All the work till now be it Data, Task definition, training the model so that Loss is minimized was on the training data. Once the model is trained on the training data, it needs to perform well on the test set as well. **So, the evaluation is never on the training data, it's always on the test data.**



	x	y
	$[2.1, 1.2, \ldots, 5.6, 7.8]$	1
	$[3.5, 6.6, \ldots, 2.5, 6.3]$	2
	$[6.3, 2.6, \ldots, 4.5, 3.8]$	3
	$[2.8, 3.6, \ldots, 7.5, 2.1]$	4
	$[2.2, 1.7, \ldots, 2.5, 1.8]$	2



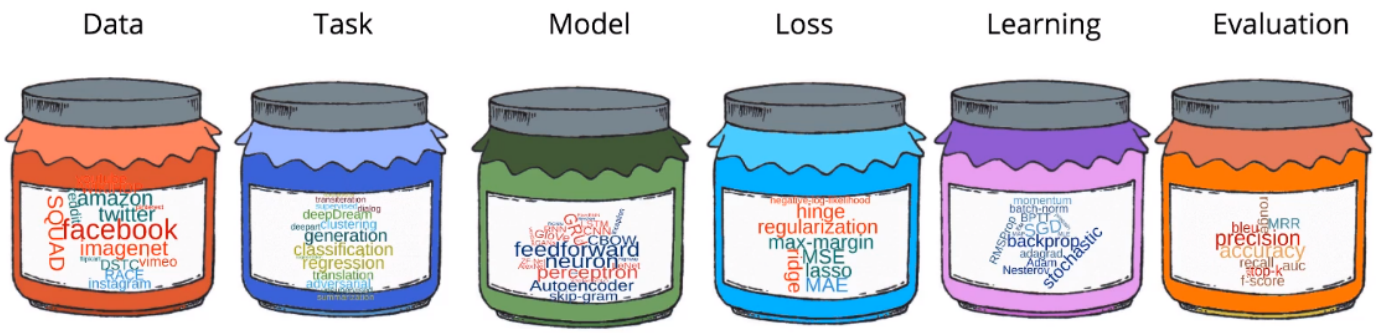
$$min_{a,b,c}\, \mathscr{L}_1 = \sum_{i=1}^{n} (y_i - \hat{f}_1(x_i))^2$$

Test Data

	x	y
	$[6.3, 2.6, \ldots, 4.5, 3.8]$	1
	$[2.8, 3.6, \ldots, 7.5, 2.1]$	3
	$[2.2, 1.7, \ldots, 2.5, 1.8]$	4

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

So, with this, we complete the jars of Machine Learning.



This article covers the content covered in the Expert Systems — 6 Jars module of the Deep Learning course and all the images are taken from the same module.

Machine Learning Padhai One Fourth Labs Deep Learning Ml Basics